Séquencement d'une ligne de montage multi-modèles : application à l'industrie du véhicule industriel

Karim Aroui

UNIVERSITÉ GRENOBLE ALPES

THÈSE

Pour obtenir le grade de

DOCTEUR DE L'UNIVERSITÉ GRENOBLE ALPES

Spécialité : **Génie Industriel**

Arrêté ministériel : 7 août 2006

Présentée par

Karim AROUI

Thèse dirigée par **Yannick FREIN** et
codirigée par **Gülgün ALPAN**

préparée au sein du **Laboratoire G-SCOP**
dans **l'École Doctorale I-MEP²**

Séquencement d'une ligne de montage multi-modèles : application à l'industrie du véhicule industriel

Thèse soutenue publiquement le **27 mai 2015**,
devant le jury composé de :

M. Patrick CHARPENTIER
Professeur, Université de Lorraine, Rapporteur
M. Lionel DUPONT
Professeur, École nationale supérieure des Mines d'Albi Carmaux, Rapporteur
M^me^ Valérie BOTTA-GENOULAZ
Professeur, Institut National des Sciences Appliquées de Lyon, Présidente
M^me^ Séverine DURIEUX
Maître de conférences, Institut Français de Mécanique Avancée, Examinateur
M. Jérôme THOMAZEAU
Volvo Group Trucks Operations, Examinateur
M. Yannick FREIN
Professeur, Grenoble INP, Directeur de thèse
M^me^ Gülgün ALPAN
Maître de conférences, HDR, Grenoble INP, Codirectrice de thèse

Université Joseph Fourier / Université Pierre-Mendès-France /
Université Stendhal / Université Savoie Mont Blanc / Grenoble INP

Table des matières

Liste des figures

Liste des tableaux

Glossaire

MMAL (Mixed-Model Assembly Line) : ligne d'assemblage multi-modèles.

Temps de cycle : le temps entre deux produits consécutifs sur la ligne.

Poste de travail : la ligne d'assemblage est divisée en postes de travail. En général, la longueur d'un poste de travail est équivalente à un temps de cycle. Plusieurs opérateurs peuvent travailler en parallèle sur le même poste.

Zone de travail : souvent, ceci correspond à un temps de cycle et donc un poste de travail. Dans certains cas, la zone de travail est plus large que le temps de cycle afin de permettre aux opérateurs de continuer leurs opérations au-delà de ce temps de cycle.

Retard : quand l'opérateur n'est pas capable de finir ces opérations avant que le produit ne sorte de sa zone de travail, un **retard** apparait. Il est appelé aussi **surcharge**.

Modèle : représente l'ensemble des produits ayant les mêmes temps opératoires.

Charge de travail d'un opérateur : c'est la somme des temps opératoires que doit effectuer l'opérateur sur tous les produits divisée par le temps disponible pour une période donnée (une journée par exemple).

VI : véhicule industriel

OTI (Order To Invoice) : service commercial central du groupe Volvo pour la marque Renault Trucks (marque des véhicules industriels assemblés à l'usine de Bourg en Bresse).

DPL (Département de Programmation Logistique) : service de planification central du groupe Volvo.

SPPP (Service Programmation et Pilotage de la Production) : service de planification de l'usine de Bourg en Bresse).

PIC : plan industriel et commercial.

PDP : programme de production.

CIA : contraintes Industrielles et d'Approvisionnement. Ce sont des limitations des nombres de variantes par jour qui modélisent les contraintes d'approvisionnement, les contraintes techniques et les contraintes d'organisation (humaines).

UEP : unité élémentaire de production.

Chapitre 1 : Introduction

Depuis plusieurs décennies, la production industrielle est soumise à une conjoncture économique incertaine caractérisée par la modification du rapport entre l'offre et la demande. Pour satisfaire les attentes des clients, les industriels sont contraints à proposer des produits de plus en plus personnalisés en multipliant les options et les fonctionnalités tout en tenant en compte des innovations technologiques, ce qui entraîne une diversité de plus en plus importante. Produire cette diversité, entrainant des variations importantes dans la durée des tâches, sur une même ligne de production, est un vrai défi dans la conception et la gestion de la ligne. D'après (Okamura and Yamashina, 1979), ce défi peut être géré par deux degrés de liberté :

- L'équilibrage des charges de la ligne qui consiste à définir les effectifs et à répartir les opérations aux différents postes de travail de la ligne. Cet équilibrage est dit équilibrage statique. Nous supposerons dans cette thèse que la ligne a été correctement équilibrée.
- Le séquencement qui consiste à déterminer l'ordre de passage des véhicules sur la ligne d'assemblage. Ce séquencement doit permettre d'obtenir ce qui est appelé un équilibrage dynamique.

L'équilibrage statique de la ligne est un problème à long/moyen terme. Il prend en compte les données prévisionnelles de planification. Tout en optimisant des objectifs divers (généralement des objectifs de rentabilité), l'équilibrage tente de réduire les effets des variations des temps opératoires d'un produit à l'autre. Cependant, ce lissage n'est généralement pas parfait puisque l'équilibrage est poussé par des objectifs de rentabilité (la rentabilité de la ligne veut dire une meilleure utilisation des ressources mobilisées, comme, par exemple, la maximisation de la charge des opérateurs) et les données prévisionnelles sur lesquelles il est basé sont parfois erronées.

Le séquencement est un problème à court terme. Il influe sur le coût de la production et conditionne sa faisabilité. Les décisions prises lors de l'équilibrage statique de la ligne sont des données d'entrée pour le problème du séquencement, dont un des objectifs est le lissage de la charge des opérateurs. Comme les postes de travail sont équilibrés en fonction du temps moyen de production, une succession de produits à forte intensité de travail pourrait conduire à une surcharge de travail des opérateurs telle qu'une tâche pourrait ne pas être terminée dans les limites du poste. Dans ce cas, la surcharge est compensée par exemple en augmentant la vitesse d'exécution des tâches ou en arrêtant la ligne.

Ce travail est effectué en collaboration avec le groupe Volvo à travers son usine d'assemblage de véhicules industriels de Bourg en Bresse. Ces véhicules sont fabriqués sur des lignes d'assemblage. Cette thèse a pour but de proposer une méthode innovante pour déterminer le séquencement des véhicules sur la ligne d'assemblage en diminuant les surcharges des opérateurs.

Deux approches peuvent être utilisées pour optimiser le lissage de charge dans un problème de séquencement : l'utilisation directe des temps opératoires ou le respect de règles. La première approche traite directement la surcharge de travail dans sa fonction objectif alors que la deuxième applique un objectif de substitution en modélisant les variations des temps opératoires par des règles qui contraignent les options les plus difficiles à réaliser en terme de temps opératoires. L'objectif est donc de respecter ces règles. Dans ce travail, nous utilisons la première approche, celle de la prise en compte directe des temps opératoires.

Un cumul de retards engendre du stress et de la fatigue et induit les opérateurs à se mettre en mauvaises postures, mais aussi à occasionner des défauts qualité à cause des oublis dus à la précipitation. L'objectif de notre étude est de diminuer ces situations critiques à travers la minimisation des retards des opérateurs.

Ce manuscrit est organisé en deux parties.

Dans la première partie nous positionnons notre problème. Elle est composée de deux chapitres.

Nous dédions le deuxième chapitre à l'étude de la littérature du problème du séquencement des lignes d'assemblage multi-modèles. Nous décrivons tout d'abord brièvement le problème d'équilibrage pour faciliter la compréhension de la problématique du séquencement. Puis, nous étudions les objectifs du séquencement et décrivons les approches utilisées pour lisser la charge. Nous détaillons ensuite l'apport de notre approche par rapport à l'existant.

Dans le troisième chapitre, nous analysons notre cas d'étude. Pour ceci, nous décrivons le processus de planification, de séquencement et d'équilibrage de charge actuel dans l'usine Volvo de Bourg en Bresse. En outre, nous analysons les limites de l'approche utilisée pour le séquencement pour mieux orienter notre proposition.

La deuxième partie de ce manuscrit détaille la contribution. Elle est composée de quatre chapitres.

En premier lieu, dans le chapitre quatre, nous argumentons le choix du critère d'optimisation qui modélise les situations déduites de l'analyse du cas d'étude.

Dans le cinquième chapitre, nous proposons un modèle en programmation linéaire mixte. Nous étudions expérimentalement la complexité du modèle que nous testons ensuite sur des instances académiques et des instances du cas d'étude.

Nous consacrons le sixième chapitre à la modélisation par programmation dynamique. Nous analysons expérimentalement sa complexité et nous le testons sur des instances académiques et des instances du cas d'étude. Une heuristique basée sur le modèle en programmation dynamique est ensuite proposée et testée.

Le chapitre sept est dédié à la proposition de trois métaheuristiques : la première utilise l'approche « algorithme génétique », la deuxième métaheuristique est celle du « recuit simulé » et la troisième est une combinaison des deux premières. Ces métaheuristiques sont testées sur des instances académiques et des instances du cas d'étude.

Enfin, nous concluons ce travail en exposant les différents résultats obtenus et en donnant des perspectives pour de prochains travaux.

Première partie : Le problème étudié

Chapitre 2 : Etude de la littérature

2.1 Introduction

L'objectif de ce chapitre est d'exposer une vision globale des travaux de recherche les plus pertinents pour notre étude et de mieux positionner notre travail par rapport à la littérature existante. Ainsi, cette revue de littérature permet d'expliciter les principaux aspects originaux de nos travaux.

Notre problématique de recherche concerne le séquencement sur une ligne d'assemblage en vue de lisser la charge. Nous présentons dans ce chapitre une revue de littérature sur le problème du séquencement. Dans un premier temps, nous expliquons les enjeux des lignes d'assemblage multi-modèles. Ensuite, nous présentons un bref état de l'art sur l'équilibrage de la ligne puisqu'il est important dans le discernement de l'objectif du séquencement. Puis, nous décrivons le séquencement, ses critères et ses approches. Enfin, nous positionnons nos travaux par rapport à la littérature existante.

2.2 Enjeux des lignes d'assemblage multi-modèles

Aujourd'hui la production se caractérise, en plus de l'accroissement de la personnalisation des produits, par la réduction des délais clients, l'amélioration continue de la qualité mais aussi par un souci de productivité de plus en plus important. Ceci engendre la nécessité de réduire les cycles de production et de tendre les flux de production.

Ce contexte met les entreprises face à des choix importants dans l'organisation des flux et des moyens de production pour avoir la flexibilité requise à la personnalisation des produits et avoir des flux tendus pour réduire les délais clients. Les lignes de production sont, par exemple, des moyens qui ont subi des modifications importantes pour s'adapter au contexte de diversification des produits.

Malgré leur manque connu de flexibilité, les lignes de production sont indispensables quand les produits sont volumineux, lourds et difficiles à manipuler. C'est le cas pour l'industrie automobile et pour la production des véhicules industriels. A l'origine, ces lignes de production ont été créées par Henry Ford dans un contexte de production grande série mono-produit. Elles ont été adaptées au contexte de diversification des produits. La ligne d'assemblage multi modèles (*Mixed-Model Assembly Line* : MMAL) est connue pour être un cas particulier de lignes de production où différentes variétés de produits à base commune sont inter-mélangées pour être assemblées sur une même ligne (Monden, 1983). Les produits se déplacent sur un moyen de transport (s'appuyant souvent sur un convoyage automatisé) leur permettant de passer par les différents postes de travail de la

ligne à vitesse constante dans un ordre prédéterminé. Le temps entre deux produits consécutifs dans la ligne est donc constant et est appelé **temps de cycle**. Les opérateurs escortent les produits tout en réalisant leurs tâches. La figure 2.1 montre la zone de déplacement de chaque opérateur dans son poste de travail.

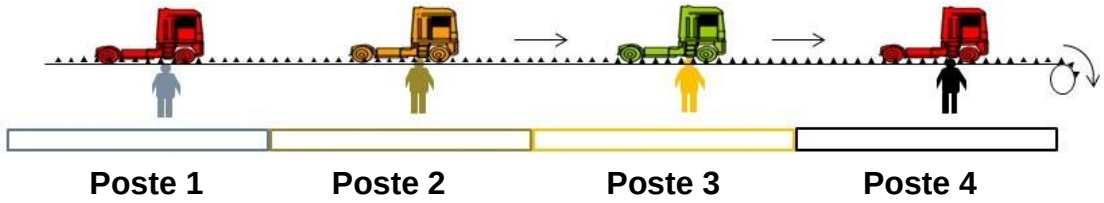


Figure 2.1 : Ligne d'assemblage multi modèles

Ces lignes MMAL sont largement utilisées, de la fabrication des produits électroniques à l'industrie automobile et plus particulièrement dans un contexte de production en Juste-à-Temps. Plusieurs travaux sont appliqués aux MMAL dans l'industrie automobile la production des véhicules industriels comme par exemple Comby (1996), Villeminot (2004) et Lesert (2006). Ce type de ligne de production permet de faire correspondre la production à l'évolution des exigences des clients tout en gardant de petites tailles de stock. Cette flexibilité est obtenue grâce au découpage des opérations et leurs affectations aux opérateurs. Deux produits successifs sur la ligne peuvent alors être très différents, mais ceci introduit des variations dans la charge de travail des ressources ce qui augmente la sensibilité des lignes de production. Dès qu'une opération ne peut pas être effectuée normalement à cause, par exemple, d'une surcharge, le fonctionnement synchronisé de la ligne impose la compensation de celle-ci ou à défaut l'arrêt de la ligne tant que cette opération n'est pas terminée. Dans ce dernier cas, un problème sur un poste de travail impacte toute la ligne, il faut donc l'éviter dans la mesure du possible, d'où l'importance de l'ordre dans lequel les produits sont fabriqués sur la ligne.

Le séquencement est la détermination de l'ordre dans lequel les produits doivent être introduits dans la ligne de montage sur un horizon de temps déterminé (souvent un jour). Il a une grande importance sur une MMAL vu que les temps opératoires dépendent des produits. Des pics de charge apparaissent quand les temps d'exécution des tâches varient considérablement d'un produit à l'autre et quand certains temps d'exécution dépassent le temps de cycle. Un des objectifs du séquencement est de fournir une liste de produits lissant la charge de travail des opérateurs. En plus, les exigences des postes peuvent être contradictoires : un même produit peut avoir un temps opératoire long dans un poste de travail sans avoir des temps opératoires longs sur le reste des postes.

Pour augmenter la flexibilité, des postes de travail, en amont de la ligne de montage, peuvent être créés pour absorber la diversité des produits. Ces postes prennent en charge

les opérations dont les temps opératoires sont très variables d'un produit à l'autre en créant des sous-ensembles prêts à assembler sur la ligne. Cependant on ne peut pas externaliser toutes les opérations à variabilité importante de la ligne d'assemblage. Il y a des opérations qui doivent être exécutées sur la ligne pour des contraintes techniques ou des contraintes de coûts. L'affectation des opérations aux différents postes de travail de la ligne, appelée équilibrage de charge, reste alors primordiale pour gérer la flexibilité de la ligne. Ce problème d'équilibrage a évolué en raison de la pression du marché qui implique une diversification accrue des produits et une stabilité moindre des quantités demandées (Giard, 2003). L'équilibrage de charge a des contraintes et des objectifs qui sont principalement des objectifs de productivité (par exemple, la minimisation du nombre d'opérateurs requis). Un mauvais équilibrage conduit à une mauvaise utilisation des ressources mobilisées et compromet la rentabilité de la chaîne. Pour que la ligne soit flexible d'un point de vue séquencement, il faut que tous les produits, peu importe leurs complexité, passent sans difficulté sur la ligne. Pour qu'elle soit rentable, il faut que les opérateurs soient chargés au maximum. En d'autres termes, pour que la ligne soit flexible, le temps de cycle doit être plus grand que le maximum des temps opératoires ce qui compromet la rentabilité puisque les opérateurs ne sont pas chargés au maximum. Il faut donc trouver le bon compromis entre la productivité et la flexibilité. Ce compromis tourne généralement en faveur de la productivité dans un contexte de concurrence accrue. C'est alors au séquencement de gérer ce manque de flexibilité : il est de plus en plus difficile de trouver une bonne séquence de produits qui diminue les surcharges des opérateurs.

2.3 L'équilibrage de charge

L'équilibrage n'étant pas l'objet de cette thèse, l'objectif de cette partie n'est donc pas de présenter un état de l'art exhaustif de cette problématique. Nous en réalisons, néanmoins, un bref aperçu puisque le séquencement est conditionné par l'équilibrage de charge. Pour une étude de littérature plus approfondie, on pourra se référer à (Becker et Scholl, 2006) et (Boysen et al., 2008).

L'équilibrage de la ligne d'assemblage est le problème de répartition des opérations aux différents postes de travail de la ligne (Becker et Scholl, 2006). Ce problème est aussi appelé ALBP (*Assembly line balancing problem*).

Pour fabriquer un produit sur une ligne d'assemblage, on divise le travail total à effectuer en un ensemble d'opérations élémentaires (appelées aussi tâches) $V = \{1, \ldots, T\}$. Chaque opération k requiert un temps opératoire τ_k. L'équilibrage consiste à répartir les opérations sur les postes de travail. Des contraintes de précédence existent entre les

opérations pour des raisons organisationnelles et techniques. Tous ces éléments peuvent être schématisés dans un graphe de précédence (figure 2.2). Chaque opération est représentée par un nœud, les poids sur les nœuds sont les temps opératoires et les arcs entre les nœuds indiquent les contraintes de précédence. Le graphe de précédence est une donnée d'entrée au problème d'équilibrage puisque les contraintes de précédence doivent être prise en compte lors de la répartition des opérations aux postes de travail de la ligne.

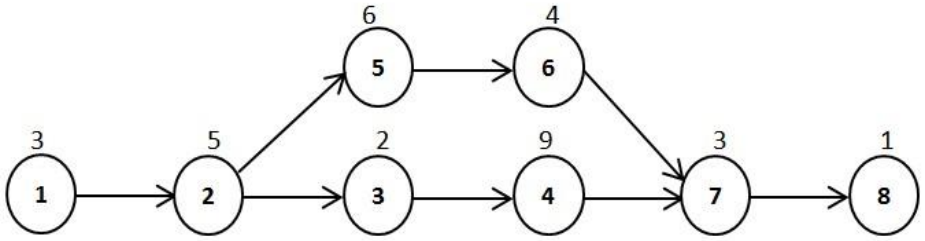


Figure 2.2 : Exemple d'un graphe de précédence

La plupart des travaux dans l'équilibrage d'une ligne d'assemblage se sont concentrés sur la modélisation et la résolution du SALBP (*simple assembly line balancing problem*). Ce problème s'applique aux lignes d'assemblage à modèle unique (Baybars, 1986) ; (Becker et Scholl, 2006). Un modèle représente l'ensemble des produits ayant les mêmes temps opératoires. Dans les lignes d'assemblage à modèle unique, chaque poste répète alors les mêmes opérations puisque tous les produits sont identiques. Dans ce cas, le temps de cycle, noté γ, est le maximum des temps opératoires des différents postes.

Les différentes versions du SALBP sont données dans le tableau 2.1. Le SALBP-F est un problème de faisabilité. Ce problème de décision détermine si une configuration d'assemblage réalisable existe pour un temps de cycle et un nombre de postes de travail donnés. Le SALBP-1 minimise le nombre de postes quand le temps de cycle est donné et le SALBP-2 minimise le temps de cycle quand le nombre de postes est donné. Le SALBP-E maximise le rendement *E* de la ligne en agissant sur le temps de cycle et le nombre de postes.

Tableau 2.1 : Versions du SALBP

		Temps de cycle	
		Donné	À minimiser
Nombre de postes	Donné	SALBP-F	SALBP-2
	À minimiser	SALBP-1	SALBP-E

Sur une ligne d'assemblage multi modèles plusieurs modèles différents sont fabriqués, ce qui engendre des différences dans les temps opératoires et dans les contraintes de

précédence. Dans ce cas, le problème d'équilibrage est une généralisation du SALBP et est appelé MALBP (*mixed assembly line balancing problem*). Pour que les opérateurs ne soient pas sous chargés, le temps de cycle est inférieur au maximum des temps opératoires sur l'ensemble des postes. Par conséquent, les temps opératoires de certains modèles dépasseront le temps de cycle. La ligne doit alors être suffisamment flexible pour pallier aux violations locales du temps de cycle.

Le problème consiste à trouver le nombre de postes de travail et le temps de cycle mais aussi un équilibrage de charges qui optimise un objectif de lissage de charge et de réduction des coûts. Les différentes versions du MALBP sont les mêmes que celles du SALBP détaillées dans le tableau 2.1.

Comme le problème de faisabilité est un problème de décision de complexité NP-complet, les versions d'optimisation du MALBP sont NP-difficiles (Wee et Magazine, 1982) ; (Scholl, 1999). Pour modéliser et résoudre le problème deux approches sont souvent utilisées : la réduction du problème en SALBP et l'équilibrage horizontal (lissage des temps attribués à chaque poste pour tous les modèles). Ces deux approches seront rapidement décrites dans la suite

L'approche de réduction du problème en SALBP est utilisée par exemple par (Roberts et Villa, 1970) en décomposant le MALBP en M problèmes SALBP indépendants (M étant le nombre de modèles). (McMullen et Frazier, 1997) prennent en compte les temps moyens des opérations pour réduire le problème en SALBP.

La variation des temps opératoires d'un modèle à l'autre pour un poste de travail d'une ligne d'assemblage multi modèles engendre des retards et des repos : un retard apparait quand l'opérateur n'est pas capable de finir ses opérations avant que le produit ne sorte de sa zone de travail ; un repos apparaît quand l'opérateur finit ses tâches sur un produit avant la fin de sa zone de travail. L'approche de l'équilibrage horizontal minimise les effets des variations des temps opératoires d'un modèle à l'autre. Cette approche minimise les retards potentiels. À cette étape de l'équilibrage, les retards sont calculés à l'aide des taux d'apparitions des modèles dans le planning de production prévisionnel. Par analogie, on peut aussi minimiser les temps de repos.

Dans l'approche de l'équilibrage horizontal, des objectifs sont proposés quand le temps de cycle et le nombre de postes sont donnés. Soit $\bar{\tau}_m$ le temps opératoire moyen pour un modèle m ($m \in 1, \dots, M$) à chacun des postes P et obtenu par $\bar{\tau}_m = \sum_{k=1}^{T} \tau_{km}/P$ (τ_{km} étant le temps opératoire de l'opération k pour le modèle m, P le nombre de postes et T le nombre d'opérations). (Thomopoulos, 1970) minimise la somme des valeurs absolue des différences entre $\bar{\tau}_m$ et leurs temps opératoires effectifs t_{mp} dans le poste p

($p = 1, \dots, P$), t_{mp} est la somme des temps des opérations τ_{km} à effectuer dans le poste p :

$$\text{Minimiser } \Psi_1 = \sum_{p=1}^{P} \sum_{m=1}^{M} \left| t_{mp} - \bar{\tau}_m \right|.$$

(Decker, 1993) quant à lui propose de minimiser la différence maximale entre les temps opératoire sur les différents modèles et pour tous les postes par rapport au temps moyen $\bar{\tau}_m$ de chaque modèle :

$$\text{Minimiser } \Psi_2 = \max_p \max_m \left\{ \left| t_{mp} - \bar{\tau}_m \right| \right\} \; avec \quad p = 1, \dots, P \;\; et \;\; m = 1, \dots, M.$$

Un troisième objectif est donné par (Domschke, 1996) qui minimise la somme des retards potentiels sur tous les modèles et tous les postes :

$$\text{Minimiser } \Psi_3 = \sum_{p=1}^{P} \sum_{m=1}^{M} \max\{0, t_{mp} - \gamma\}.$$

(Domschke, 1996) mène une étude comparative de ces trois objectifs qui montre que le troisième objectif est le plus performant pour anticiper et éviter les effets de variation des temps opératoires.

Le GALBP (generalised assembly line balancing problem) est une généralisation du MALBP. Plusieurs auteurs ont, en effet, proposé de prendre en compte des critères supplémentaires d'optimisation en plus de ceux considérés dans le MALBP. (Pastor et al., 2002) considèrent le problème du MALBP-2 avec un objectif supplémentaire qui essaye d'augmenter l'uniformité des opérations dans les postes de travail. (Vilarinho et Simaria, 2002) essayent d'assurer un équilibrage horizontal (pour chaque poste, les temps opératoires des différents modèles doivent être lissés) et un équilibrage vertical (les charges de travail des différents postes doivent être lissées entre elles) en même temps.

2.4 Le séquencement

Le problème du séquencement de la ligne d'assemblage consiste à trouver la séquence de production (l'ordre de passage sur la ligne) d'un nombre donné de produits sur un horizon de planification (en général une journée). C'est un problème à court terme. Les décisions prises à l'équilibrage de charge (problème à long-moyen terme) sont des données d'entrée pour le problème du séquencement. Dans ce travail, nous considérons que l'équilibrage de la ligne est déjà effectué et nous nous concentrons sur le problème de séquencement.

Malgré le fait que chaque combinaison de produits soit une séquence « envisageable », l'impact économique et social du séquencement oblige à l'étudier d'une façon minutieuse. Les deux critères généraux d'optimisation du problème de séquencement sont

liés à la charge des opérateurs et la demande de pièces et composants en bord de ligne (Bard et al., 1992).

- *Lissage de la consommation des pièces*

La variation des modèles et des options engendre l'utilisation de pièces et de composants différents d'un produit à l'autre sur la ligne. L'ordre de passage des produits sur la ligne influe alors sur l'approvisionnement des pièces gérées en juste-à-temps. Le juste-à-temps impose que les pièces soient livrées au bon endroit et au bon moment sans gaspillage. Donc la séquence doit faciliter ceci, en rendant la consommation des pièces et composants la plus régulière possible tout au long de l'horizon de planification. Dans le cas contraire, des stocks de sécurité seront nécessaires pour affronter les pics de demande ce qui est à l'encontre des objectifs du Juste-à-temps. C'est pour cela que ce critère est aussi appelé « critère du Juste-à-temps ».

- *Lissage de la charge de travail*

La personnalisation de l'offre de produits engendre des variations dans les temps opératoires des postes. Un produit avec option nécessite un temps opératoire plus important qu'un produit sans option. Si plusieurs produits nécessitant des temps opératoires importants pour un poste de travail se succèdent, des pics de travail peuvent apparaître ce qui engendre, par exemple, l'utilisation d'opérateurs polyvalents pour compenser les surcharges ou, dans le cas d'une surcharge trop importante, un arrêt de la ligne. L'ordre de passage des véhicules peut éviter ces situations en définissant une séquence où ces produits à temps opératoires importants alternent avec des produits dont les temps opératoires sont moins importants (Boysen et al., 2009a).

Quelques auteurs ont considéré les deux critères (lissage de la charge et lissage de la consommation des pièces) simultanément (on peut citer, entre autres, (Aigbedo et Monden, 1997) et (Kotani * et al., 2004). Un état de l'art détaillé du problème de séquencement est donné par Boysen et al. (2009). Nous allons donner les principaux éléments des deux approches dans les deux sections suivantes.

2.5 L'approche utilisée pour lisser la consommation des pièces

L'approche appliquée pour lisser la consommation des pièces est connue sous le nom de « *level scheduling* ». Cette approche permet de faciliter l'approvisionnement des pièces en juste-à-temps et de minimiser les stocks de sécurité. Un taux de consommation théorique est attribué à chaque pièce. Ce taux est la division de la demande totale de la

pièce par l'horizon de temps du séquencement. L'approche cherche à trouver la séquence qui permet d'avoir un taux réel de consommation aussi proche que possible du taux théorique. Considérons M un ensemble de modèles à séquencer, chaque modèle i utilise différentes pièces s ($s \in S$). Le taux théorique de consommation se calcule comme suit :

$$z_s = \frac{\sum_{i \in M} h_{is} \,.\, n_i}{N} \qquad \forall s \in S$$

Avec N le nombre total de produits de la séquence, n_i la demande du modèle i et h_{is} le nombre d'unités de la pièce s consommée pour la production d'un modèle i.

La fonction objectif est alors de minimiser la déviation du taux réel par rapport au taux théorique de consommation (par exemple (Monden, 2011) ; (Bautista et al., 1996)). y_{ij} est le cumul de production du modèle i jusqu'à la position j dans la séquence :

$$\text{Minimiser } Z = \sum_{j=1}^{N} \sum_{s \in S} (\sum_{i \in M} h_{is} \,.\, y_{ij} - j \,.\, z_s)^2$$

Afin de lisser la consommation des pièces on peut aussi lisser la production des modèles (Giard et Jeunet, 2004). Cette variante du problème est appelée « Product Rate Variation » (PRV). D'après les adeptes de cette variante du problème, lisser la production des modèles revient à lisser la consommation des pièces, lorsque les produits finis utilisent tous le même ensemble de composants (Miltenburg, 1989) ou lorsque les pièces requises pour la production de chaque modèle sont différentes (Kubiak, 1993). Cependant, (Boysen et al., 2009b) montrent que ces hypothèses sont inappropriées au cas réel et actuel des lignes de montage multi-modèles.

La figure 2.3 montre la droite de lissage idéal pour un modèle i et sa production réelle. Le but est de minimiser la distance entre la production effective et la production idéale du modèle.

Différentes méthodes ont été utilisées pour résoudre les deux variantes du problème (lissage de la consommation des pièces et le lissage de la production des modèles). Par exemple, (Monden, 1983) propose une heuristique myope connue sous le nom de Goal Chasing Method. (Bautista et al., 1996) améliorent l'heuristique de Monden et proposent une méthode exacte basée sur la programmation dynamique. Plusieurs heuristiques et une méthode d'affectation exacte basée sur un algorithme pseudo-polynomiale (avec une complexité de (N^3), N étant le nombre total des produits de la séquence) sont proposées par (Kubiak, 1993) pour la variante PRV. La complexité est réduite à $O(N\,Log\,N)$ par (Khadka et Werner, 2013) qui proposent également une procédure exacte basée sur la programmation non linéaire en nombres entiers pour le même problème avec la pondération des déviations.

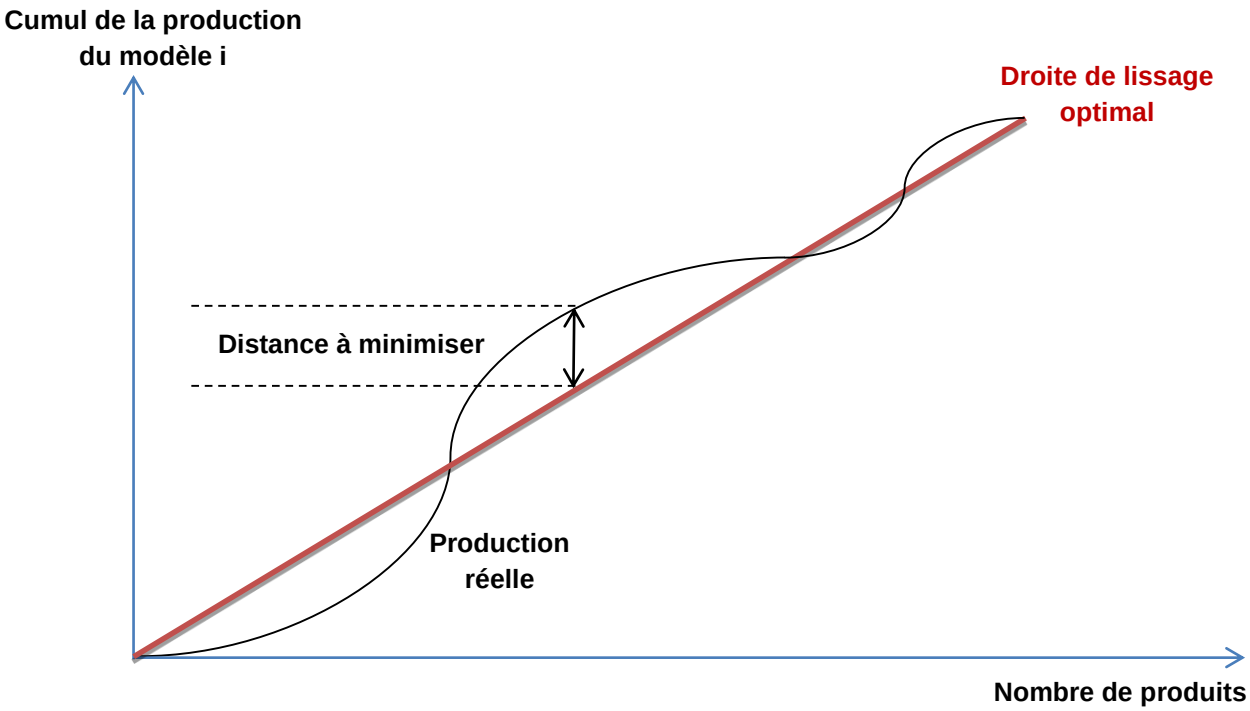


Figure 2.3 : Lissage de la production des modèles

2.6 Les approches utilisées pour lisser la charge de travail

Le critère lissage de la charge de travail a été repris dans deux approches de séquencement données dans (Boysen et al., 2009a) :

- *mixed-model sequencing* : le but est de minimiser la surcharge de travail en prenant en compte explicitement les temps opératoires ;
- *Car sequencing* : ces problèmes tentent de minimiser la surcharge de travail de manière implicite afin d'éviter l'effort de collectionner les données liées à l'approche *mixed-model sequencing*. Ceci passe par la minimisation du nombre de violations d'un ensemble de règles de séquencement (Boysen et Fliedner, 2006) ; (Golle et al., 2010) ;

2.6.1 Le car sequencing

Dans l'approche *car sequencing*, un ensemble d'options de produit est pris en compte pour la définition des règles de séquencement (Golle et al., 2010). Ces règles consistent en des contraintes d'espacement c'est-à-dire un maximum d'occurrences possibles d'une option (ou combinaison d'options) dans une sous-séquence d'une certaine longueur. Le problème du *car sequencing* a pour but de trouver une séquence qui ne viole pas ces contraintes. Le problème découle des applications de l'industrie automobile (Parrello,

1988); (Nguyen et Cung, 2005). Il a été formulé pour la première fois par (Parrello, 1988).

Une contrainte d'espacement N_o/P_o signifie que dans une sous-séquence de P_o produits successifs, on ne doit pas avoir plus de N_o produits répondant au critère o. Le critère o représente une option ou une combinaison d'options. Ces contraintes reflètent les contraintes de capacité dans les postes de travail (Drexl et Kimms, 2001).

La figure 2.4 schématise un exemple de règle de séquencement : une contrainte d'espacement de 1/5 régissant l'option A. Dans une sous-séquence de 5 produits, on ne doit pas avoir plus de 1 produit avec l'option A. Dans l'exemple, on viole deux fois cette contrainte. En plus des contraintes d'espacement, d'autres types de contraintes sont utilisés dans l'approche du *car sequencing* comme, par exemple, les contraintes de succession (Boysen et al., 2009a) qui sont des règle de type « un produit avec une option A ne doit pas succéder à un produit avec une option B ».

Le *car sequencing* peut être formulé en un problème de satisfaction de contraintes grâce aux règles de séquencement (voir (Brailsford et al., 1999) ou bien en un problème d'optimisation. Dans le deuxième cas, le critère d'optimisation est la minimisation du nombre de viols de contraintes (Gottlieb et al., 2003) ; (Gravel et al., 2005).

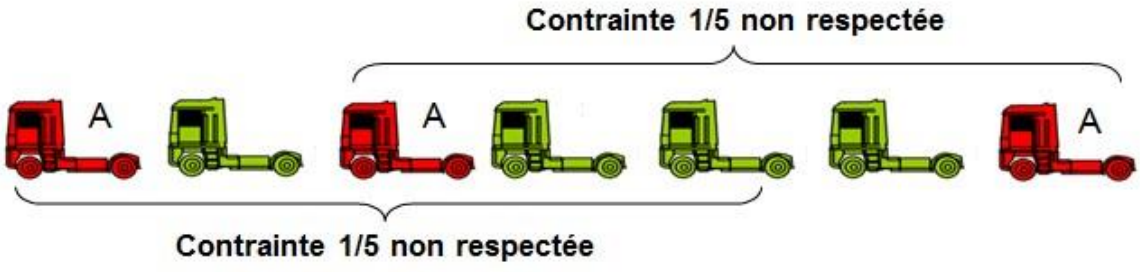


Figure 2.4: Exemple d'une contrainte d'espacement

Dans des cas réels d'application (par exemple le constructeur automobile Renault), on différencie les contraintes « fortes » et les contraintes « molles » selon l'élasticité ou non de la capacité du poste de travail d'où ces contraintes sont originaires (Nguyen et Cung, 2005) ; (Gagné et al., 2006) ; (Solnon et al., 2008). Ainsi, les postes capacitaires sont ceux dont la capacité est fixe alors que les postes non-capacitaires peuvent voir leur capacité s'accroître en accueillant des travailleurs supplémentaires. Les contraintes « fortes » sont liées à des postes capacitaires : le non-respect d'une de ces contraintes arrêtera la ligne d'assemblage et provoquera une perte de productivité. Les contraintes « molles » sont liées à des postes non capacitaires : le non-respect d'une de ces contraintes provoquera l'intervention d'un opérateur polyvalent engendrant un surcoût (Giard et Jeunet, 2004). Les deux types de contraintes peuvent être distingués par une notion de priorité.

La plupart des travaux, notamment ceux développés dans le contexte automobile, considère l'approche dite *car sequencing*. Malgré sa popularité, cette approche présente un inconvénient majeur : les règles de séquencement ne sont pas triviales à définir. Au niveau industriel, il n'existe pas de méthode claire pour la définition des contraintes. Elles sont généralement définies d'une façon empirique. (Lesert, 2006) montre que dans son cas d'étude, le choix des contraintes d'espacement s'effectue en fonction des difficultés remontées de l'atelier et de l'expérience des équilibreurs et non pas en se basant sur une étude des temps opératoires. En plus, la définition d'une contrainte se base parfois sur une combinaison compliquée d'options, ce qui rend la tâche plus difficile. Enfin, la définition des contraintes se base sur les données prévisionnelles qui s'écartent généralement de la production réelle.

En conséquence, on peut observer dans de nombreux cas une absence de corrélation entre respect de contrainte et pic de charge. Les contraintes d'espacement sont définies pour lisser les pics de charge des opérateurs. Le respect de ces contraintes d'espacement devrait donc permettre d'éviter qu'un opérateur ne se retrouve en surcharge. Pour vérifier cet élément, (Lesert, 2006) mesure la qualité des séquences selon deux points de vue : celui du respect des contraintes et celui de la charge des opérateurs. Pour le premier point de vue, la qualité de la séquence est mesurée en fonction du nombre de non respects de contraintes. Pour le deuxième point de vue, elle est mesurée en fonction du nombre de fois qu'un opérateur polyvalent a été sollicité (ce nombre est aussi appelé nombre d'alertes). Ces deux mesures devraient être corrélées : le respect des contraintes doit conduire à un nombre très faible d'alertes. Dans le cas d'étude de (Lesert, 2006), la séquence respecte très bien les contraintes d'espacement mais le nombre moyen d'alertes quotidiennes est très élevé. Cette incohérence est liée au fait que les contraintes ne parviennent pas à saisir la vraie surcharge de travail à minimiser et à une défaillance dans la qualification d'un pic de charge (Lesert et al., 2011).

Notons enfin une difficulté pour prioriser les contraintes : dans le cas des séquences où le respect de toutes les contraintes n'est pas possible, il est difficile d'attribuer des priorités aux contraintes d'une façon méthodique.

L'avantage du *car sequencing* est principalement le volume des données à utiliser qui est moins important que dans l'approche « *mixed-model sequencing* » qui nécessite de prendre explicitement tous les temps opératoires.

2.6.2 Le mixed model sequencing

Lors de l'équilibrage de charge d'une ligne de montage multi-modèles, malgré les différences entre les temps opératoires des modèles, on ne peut pas prendre un temps de

cycle égal au plus grand temps opératoire car les taux de charge des postes de travail seraient très faibles. Pour un poste de travail donné, on a donc des modèles dont le temps opératoire est supérieur au temps de cycle et d'autres dont le temps opératoire est inférieur au temps de cycle. La succession de modèles avec des temps opératoires élevés engendre un retard pour l'opérateur concerné. Nous préciserons ci-dessous comment ces retards sont traités. Le but de cette approche est de trouver une séquence de produits permettant d'éviter ces retards en considérant explicitement les temps opératoires de chaque modèle sur chaque poste de travail (mais aussi les caractéristiques opérationnelles des postes de travail). Cette approche a été formulée pour la première fois par (Wester et Kilbridge, 1964).

Le retard est calculé selon la définition des limites du poste de travail :

- Les limites de poste coïncident avec le temps de cycle (le temps de cycle peut aussi exprimer la distance entre deux produits consécutifs sur la ligne). Les opérateurs n'ont pas le droit de travailler en dehors du temps de cycle. Dans ce cas, le retard est calculé par rapport au temps de cycle.

- On ajoute à chaque poste un peu de flexibilité en lui attribuant une zone de poste plus grande que le temps de cycle. Dans ce cas, des produits dont le temps opératoire est plus grand que le temps de cycle n'engendrent pas forcément des retards. Le retard est calculé par rapport à la nouvelle limite de poste. La figure 2.5 illustre ce cas : le premier produit n'induit pas de retard même si son temps opératoire dépasse le temps de cycle. Un retard est observé au passage du deuxième produit. Il peut affecter la date de début des opérations sur le poste suivant dans la ligne si le lien entre les postes est pris en considération (Bautista et Cano, 2011).

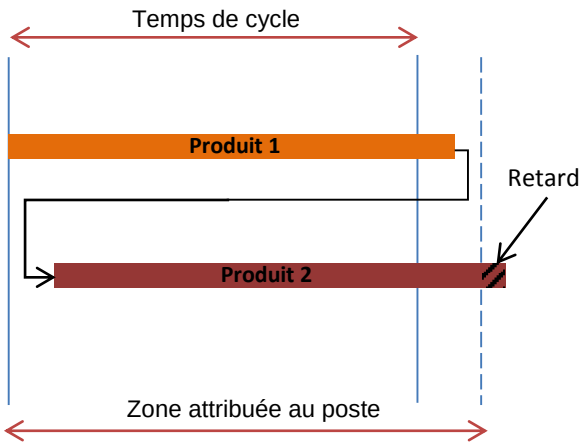


Figure 2.5 : Cas où les limites sont plus grandes que le temps de cycle

Suite à un retard (une surcharge) d'un opérateur, différentes réactions sont possibles (Boysen et al., 2009a) :

- La vitesse des opérations est accélérée afin de traiter la surcharge et les finir avant la limite de fin de poste, soit par l'opérateur lui-même, soit en demandant l'aide d'un opérateur polyvalent (Bautista et Cano, 2008).
- Les tâches non finies à la limite de fin de poste sont laissées pour d'autres opérateurs à la fin de la ligne de montage (Kim et Jeong, 2007) ; (Bautista et al., 2012).

  Pour ces deux premières réactions, la surcharge du poste concerné, si elle est calculée par rapport au temps de cycle, n'affecte pas le temps de début des opérations pour le poste suivant dans la ligne.
- La ligne est stoppée dès qu'une surcharge apparait ce qui engendre un temps de repos pour les autres postes. Ceci est le cas pour le constructeur automobile japonais Toyota. (Xiaobo et Ohno, 2000) considèrent ce type de réaction et minimisent la durée pendant laquelle la ligne est stoppée.
- L'opérateur doit finir ses tâches même s'il dépasse sa limite de poste. Par ailleurs, dans certains travaux (Goldschmidt et al., 1997), il peut même prendre de l'avance en travaillant sur un produit avant qu'il ne soit disponible dans sa zone.
- Dans quelques rares travaux (Bock et al., 2006), les produits impactant une surcharge de l'opérateur sont retirés de la ligne pour être finis ailleurs. Les postes suivants dans la ligne principale n'auront pas d'opérations à effectuer sur ces produits.

On distingue donc deux types de postes : les postes de travail fermés ou ouverts. Les opérateurs des postes de travail fermés ne peuvent pas continuer leurs tâches en dehors de la zone de travail. Dans le cas contraire, les postes de travail sont dits ouverts.

Une partie de la revue de la littérature de (Boysen et al., 2009a) est consacrée à l'approche de *mixed model sequencing*. Les auteurs proposent une classification des références et donne leurs contributions et les méthodes de résolution utilisées. Nous reprenons ces références et nous rajoutons les références publiées depuis cet état de l'art dans le tableau 2.2. Pour chaque référence, on fournit : la méthode de résolution utilisée, méthodes réparties en méthodes exactes et méthodes heuristiques, le critère d'optimisation choisi et des commentaires sur les autres aspects considérés dans l'article. Le problème du séquencement peut être couplé avec le problème d'équilibrage. Ceci est indiqué dans la deuxième colonne du tableau 2.2.

Les critères d'optimisation adoptés par les auteurs sont variés. La majorité des travaux minimise les retards ou les surcharges. Par analogie, d'autres travaux, minimise les repos

puisqu'ils représentent des ressources inutilisées. Nous rappelons qu'un repos survient quand l'opérateur finit ses tâches sur un produit et qu'il doit attendre l'arrivée du produit suivant dans sa zone de travail. En outre, quelques travaux minimisent les arrêts de la ligne qui apparaissent généralement quand d'importantes surcharges surviennent. Le dernier critère utilisé est la longueur de la ligne. Ce critère est pris en compte quand les limites des postes sont définies pendant le séquencement (cette décision est traditionnellement prise lors de l'équilibrage). Il substitue la minimisation des coûts d'investissement du système de transport de la ligne. Certains travaux, comme par exemple (Javadi et al., 2008), ont couplé ces objectifs avec d'autres critères tels que la minimisation des temps de réparation et le lissage de la consommation des pièces.

La programmation linéaire est la méthode la plus utilisée pour résoudre à l'optimale les problèmes étudiés par les travaux cités dans le tableau 2.2. Les auteurs ont aussi développé des méthodes approchées pour résoudre ces problèmes. Les métaheuristiques (essentiellement le recuit simulé et l'algorithme génétique) sont les méthodes les plus utilisées dans cette catégorie.

Les auteurs prennent en compte différentes caractéristiques des lignes d'assemblage dans la définition de leurs problèmes. Certains auteurs considèrent des postes de travail ouverts (les opérateurs peuvent effectuer des tâches au-delà des limites de leurs postes) comme par exemple (Bard et al., 1992) et d'autres les considèrent comme postes fermés.

Quelques travaux, comme par exemple (Hamzadayi et Yildiz, 2012, 2013), étudient le séquencement des lignes en U. Ces lignes permettent aux opérateurs de travailler sur plus d'un produit par temps de cycle à différentes positions de la ligne, car les postes de croisement ont accès simultanément à deux branches de la ligne en U. C'est pour cette raison que le problème de séquencement est spécifique pour ce type de lignes.

Les limites de cette approche est la taille importante des données par rapport au *car sequencing* puisqu'elle prend en compte les temps opératoires de chaque produit pour chaque opérateur. Ceci pourrait constituer un frein à l'optimisation et à la mise en place industrielle si de telles données sont difficiles à collecter.

	Equilibrage + séquencement	Méthode utilisée									Critère d'optimisation				Autres aspects
		Méthodes exactes				Méthodes approchées									
							Métaheuristiques								
		Branch & Bound	Prog Dynamique	Prog Linéaire	Autres	Heuristiques	Recuit simulé	Recherche tabou	Algorithme génétique	Autres	Arrêts de la ligne	Retards	Temps de repos	Longueur de la ligne	
(Akgündüz et Tunalı, 2009)									x			x			Multi-objectifs (temps de préparation, retards, lissage consommation des pièces)
(Bard et al., 1992)				x										x	Postes de travail ouverts
(Bautista et al., 2012)				x								x			Liens entre les postes de travail
(Bautista et Cano, 2008)						x						x			
(Bautista et Cano, 2011)				x		x						x			Interruption conditionnée des opérations suite à un retard
(Bolat, 1997)							x					x			Temps de retour à la limite sup du poste non négligeables
Bolat et Yano (1992)					x	x						x			Un poste de travail et deux modèles
(Bolat et Yano, 1992b)			x			x						x			
(Bolat et al., 1994)					x	x						x			Minimise les coûts de préparation en plus des retards
(Boysen et al., 2010)		x				x						x			Minimiser le nombre de retards
(Cano-Belmán et al., 2010)						x						x			Scatter search
(Celano et al., 2004)										x	x				Ligne en U
(Chutima et al., 2003)										x				x	Temps opératoires stochastiques
(Dar-El, 1978)						x								x	Postes de travail ouverts et fermés
(Dar-El et Cother, 1975)						x								x	Postes de travail ouverts
(Dar-El et Cucuy, 1977)						x								x	Un seul poste de travail
(Dar-El et Nadivi, 1981)						x								x	Postes de travail ouverts
(Dong et al., 2014)	x						x					x			Ligne en U – Temps opératoires stochastiques

Tableau 2.2 : Revue des références traitant le mixed model sequencing 1/3

	Equilibrage + séquencement	Méthode utilisée									Critère d'optimisation				Autres aspects
		Méthodes exactes				Méthodes approchées									
							Métaheuristiques								
		Branch & Bound	Prog Dynamique	Prog Linéaire	Autres	Heuristiques	Recuit simulé	Recherche tabou	Algorithme génétique	Autres	Arrêts de la ligne	Retards	Temps de repos	Longueur de la ligne	
(Fattahi et Salehi, 2009)							x					x	x		Temps de cycle variable
(Goldschmidt et al., 1997)						x								x	Un seul poste de travail
(Golle et al., 2014)				x								x			
(Hamzadayi et Yildiz, 2012)	x						x		x			x			Ligne en U
(Hamzadayi et Yildiz, 2013)	x						x						x		Ligne en U
(Hwang et Katayama, 2009)	x								x			x			Ligne en U et ligne classique (droite)
(Javadi et al., 2008)						x						x			Multi-objectifs (temps de préparation, retards, lissage consommation des pièces)
(Kara, 2008)	x						x								Ligne en U
(Kim et al., 1996)									x					x	Postes de travail ouverts et fermés
(Kim et Jeong, 2007)		x				x						x			Heuristique basée sur une borne inférieure et sur la recherche locale
(Mosadegh et al., 2012)	x			x			x					x			
(Okamura et Yamashina, 1979)						x					x				Temps de retour à la limite sup du poste non négligeables
(Özcan et al., 2010)	x								x						Ligne en U – Temps opératoires stochastiques
(Cortez et Costa, 2014)				x						x					Opérateurs hétérogènes
(Rabbani et al., 2008)									x			x			Multi-objectifs (temps de préparation, retards, lissage consommation des pièces)

Tableau 2.2 : Revue des références traitant le mixed model sequencing 2/3

	Equilibrage + séquencement	Méthode utilisée									Critère d'optimisation				Autres aspects
		Méthodes exactes				Méthodes approchées									
							Métaheuristiques								
		Branch & Bound	Prog Dynamique	Prog Linéaire	Autres	Heuristiques	Recuit simulé	Recherche tabou	Algorithme génétique	Autres	Arrêts de la ligne	Retards	Temps de repos	Longueur de la ligne	
(Rahimi-Vahed et al., 2007)						x						x			Multi-objectifs (temps de préparation, retards, lissage consommation des pièces)
(Rahimi-Vahed et Mirzaei, 2007)										x		x			Multi-objectifs (temps de préparation, retards, lissage consommation des pièces)
(Saif et al., 2014)	x			x								x			Métaheuristique utilisée: Colonie d'abeilles
(Sarker et Pan, 1998, 2001)						x						x	x		Postes de travail ouverts
(Scholl, 1999)	x				x					x					
(Scholl et al., 1998)							x					x			
(Sumichrast et al., 2000)										x		x			
(Tavakkoli-Moghaddam et Rahimi-Vahed, 2006)										x		x			Multi-objectifs (temps de préparation, retards, lissage consommation des pièces)
(Thomopoulos, 1970)	x					x						x	x		Postes de travail ouverts
(Tsai, 1995)					x						x	x			Un seul poste de travail
(Wester et Kilbridge, 1964)						x						x	x		Un seul poste de travail
(Xiaobo et Ohno, 1994)					x						x				La ligne est stoppée dès qu'un retard apparaît.
(Xiaobo et Ohno, 1997)		x								x	x				La ligne est stoppée dès qu'un retard apparaît.
(Xiaobo et Ohno, 2000)						x					x				La ligne est stoppée dès qu'un retard apparaît.
(Yano et Rachamadugu, 1991)					x	x						x			Deux modèles
(Yoo et al., 2005)							x	x			x		x		Temps de retour à la limite sup du poste non négligeables

Tableau 2.2 : Revue des références traitant le mixed model sequencing 3/3

2.7 Positionnement de notre problématique

Notre étude relève du problème du séquencement avec l'objectif du lissage de la charge de travail. Nous allons utiliser l'approche *mixed-model sequencing* vu les limites de l'approche *car sequencing*. Nous nous concentrons sur le problème du séquencement sur une ligne d'assemblage multi modèles avec minimisation de la surcharge de travail appelé aussi MMSP-W (*Mixed Model Sequencing Problem with Workoverload minimisation*).

La revue de la littérature proposée par (Boysen et al., 2009a) pointe les lacunes des études de l'approche du *mixed-model sequencing*. Ces lacunes sont la non-prise en compte des temps opératoires stochastiques, des agencements spécifiques des lignes de montage et de l'hétérogénéité des caractéristiques des opérateurs. Le tableau 2.2 montre que la littérature a depuis essayé de combler certaines de ces lacunes : par exemple (Özcan et al., 2011) et (Dong et al., 2014) ont considéré des temps opératoires stochastiques et des agencements spécifiques comme les lignes en U prises en compte par (Hamzadayi et Yildiz, 2012, 2013) et (Kara, 2008) entre autres. Le tableau 2.2 montre aussi que la majorité des articles proposent des heuristiques ou des métaheuristiques pour résoudre ces problèmes.

Nous constatons encore un manque de références pour la troisième lacune qui est l'hétérogénéité des caractéristiques des opérateurs. (Cortez et Costa, 2014) abordent cette hétérogénéité en proposant un modèle mathématique considérant la différence de vitesse d'exécution des tâches entre les opérateurs polyvalents et opérateurs réguliers. Ils proposent un programme linéaire mixte et des heuristiques. Cependant, ces méthodes sont testées sur des instances de petites tailles (entre 4 et 7 opérateurs). Le programme linéaire mixte ne donne pas de solution optimale au bout de deux jours de calcul mais les heuristiques donnent des solutions satisfaisantes. Notre travail a pour originalité d'aborder cette hétérogénéité des opérateurs en considérant trois types d'opérateurs ayant des types de tâches différentes. Par ailleurs, nous avons pour ambition de tester des méthodes de résolution sur des instances de tailles réelles.

Ce travail est basé sur un cas d'étude industriel qui est le centre de Bourg en Bresse de montage des véhicules industriels du groupe Volvo Group Trucks Operations. Nous utiliserons trois méthodes pour modéliser et résoudre ce problème : la programmation linéaire, la programmation dynamique et les métaheuristiques.

Plusieurs réactions possibles en cas de retard de l'opérateur ont été détaillées dans la section 2.6.2. Dans notre travail, nous considérons la première réaction : la vitesse des opérations est accélérée pour les finir avant la limite de fin de poste, soit par l'opérateur

lui-même, soit en demandant l'aide d'un opérateur polyvalent. La zone attribuée à chaque poste de travail correspond au temps de cycle de la ligne. Le retard est donc calculé par rapport au temps de cycle.

Pour conclure, les principales originalités de ce travail sont :

- L'utilisation directe des temps opératoires pour séquencer la ligne d'assemblage
- La prise en compte de trois types d'opérateurs
- Tester des méthodes exactes sur des instances industrielles

2.8 Conclusion

Dans ce chapitre, nous avons détaillé l'état de l'art relatif au problème de séquencement. Nous avons présenté le problème de l'équilibrage dont les résultats sont des données d'entrée pour le séquencement. Ensuite, nous avons énoncé les deux objectifs du séquencement et les approches de l'objectif du lissage de la charge de travail (figure 2.6). Nous avons mis en exergue les limites de l'approche *car sequencing* et nous avons présenté les travaux de l'approche *mixed-model sequencing*. Enfin, nous avons positionné notre recherche par rapport à l'existant.

Dans le prochain chapitre, nous décrivons en détails le cas industriel du centre de montage de véhicules industriels de Bourg en Bresse, sur lequel est basée notre recherche.

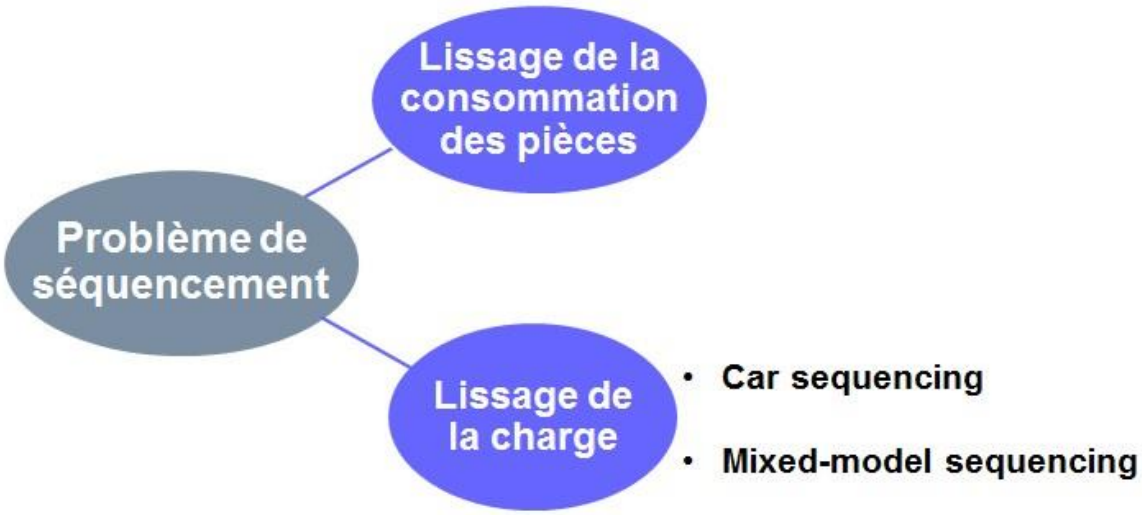


Figure 2.6 : Le problème de séquencement

Chapitre 3 : Description et analyse du cas industriel

3.1 Introduction

Notre cas d'étude est l'usine d'assemblage de véhicules industriels de Bourg en Bresse qui appartient au Groupe Volvo Group Trucks Operations. Le groupe est spécialisé principalement dans la fabrication des véhicules industriels et des équipements de construction. Les véhicules industriels sont des véhicules lourds dont le Poids Total Autorisé en Charge (PTAC) est supérieur à 3,5 tonnes. Le site de Bourg en Bresse est composé de trois lignes d'assemblage (deux lignes principales et une ligne de développement de nouveaux produits). Il est spécialisé dans la fabrication des véhicules industriels de plus de 16 tonnes de PTAC. Il est caractérisé par des cadences de production plus faibles que l'industrie automobile (facteur 1/10 environ) et par une diversité plus importante (facteur 10 environ, (Comby, 1996). Ces produits ont des coûts élevés et doivent être adaptés aux besoins du client.

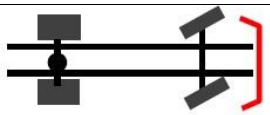
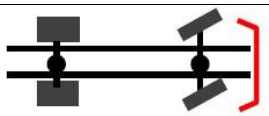
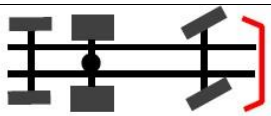
Ce chapitre a pour objet la description détaillée du problème industriel à l'origine de nos travaux de recherche. Nous présentons d'un point de vue pratique les processus de planification et nous insistons sur la procédure du séquencement et ses contraintes liées à notre contexte industriel. Nous présentons aussi l'équilibrage de charge dans notre contexte industriel puisqu'il est fortement lié au séquencement.

3.2 Le processus de planification

L'ensemble des véhicules fabriqués par le site de Bourg en Bresse est décomposé en un nombre restreint de types de produit qui sont définis par trois critères :

- Le fait que le véhicule soit porteur ou tracteur. Un véhicule tracteur tracte la charge alors qu'un véhicule porteur la porte.
- La silhouette : le nombre total de roues et le nombre de roues motrices (ce qui correspond aussi au nombre des ponts et d'essieux). Le tableau 3.1 schématise quelques exemples de silhouettes : le premier chiffre indique le nombre total de roues et le deuxième présente le nombre de roues motrices.
- Le type de motorisation.

Tableau 3.1 : Exemples de silhouettes

Silhouettes	4x2	4x4	6x2
			

La définition complète d'un véhicule donné est ensuite effectuée pour un type de produit par la définition de toutes les options (appelées aussi variantes) demandées par le client (variantes catalogue ou hors-série).

Le processus de planification passe par plusieurs acteurs de l'entreprise :

- OTI (*Order To Invoice*) : Service commercial central du groupe Volvo pour la marque Renault Trucks (marque des véhicules industriels assemblés à l'usine de Bourg en Bresse). Ce service est basé à Lyon et a pour tâches : la programmation commerciale (calcul des prévisions des demandes clients), le positionnement des commandes sur des journées de production, la négociation avec le réseau de distribution et les filiales (à propos des délais de livraison)…
- DPL (Département de Programmation Logistique) : Service de planification central du groupe Volvo. Basé à Lyon, ce service a pour tâches : la planification générale de la production (calcul des besoins, affectation des volumes de production aux usines…), la coordination entre les différents sites de production…
- SPPP (Service Programmation et Pilotage de la Production) : Service de planification de l'usine de Bourg en Bresse). Ce service a pour tâches : la programmation et la planification sur les lignes de montage et la gestion des approvisionnements des principaux composants des véhicules industriels (organes).

La figure 3.1 présente le processus global de planification pour l'usine de Bourg en Bresse. Chaque acteur est identifié par une couleur pour spécifier ses actions.

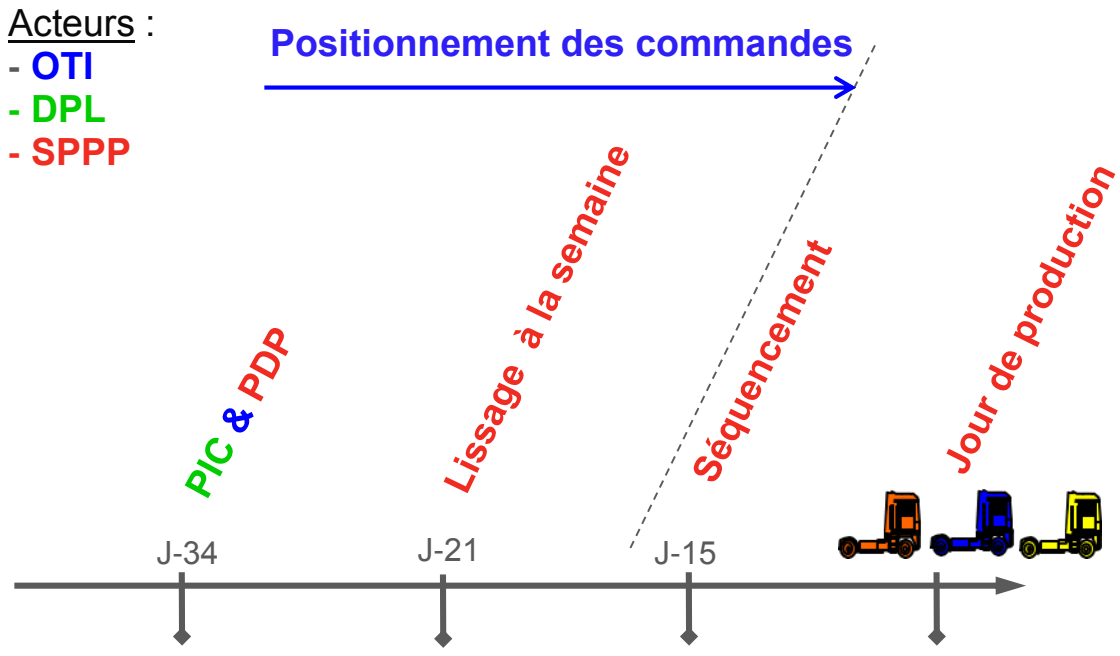


Figure 3.1 : Processus de planification du cas industriel

Chaque mois, les trois acteurs cités ci-dessus mettent en place le plan industriel et commercial et le programme de production. Ensuite, un lissage à la semaine est effectué par SPPP afin d'avoir une production équilibrée d'une journée à l'autre d'une même semaine. Les commandes peuvent être positionnées tout au long du processus de planification jusqu'avant le séquencement. Ce dernier est effectué environ 15 jours avant le jour de production.

Dans la suite, nous présentons les différentes étapes de la planification avec plus de détails.

3.2.1 Plan industriel et commercial & programme de production

Chaque mois, tous les acteurs cités ci-dessus mettent à jour le plan industriel et commercial et le programme de production sur l'horizon *M+2* jusqu'à *M+12* (les commandes des mois *M* et *M+1* étant déjà figées) : le service commercial fournit les données prévisionnelles de ventes au service de planification générale (DPL) qui calcule les besoins et transmet les volumes à produire par mois aux services de planification de toutes les unités de production (dont SPPP Bourg) qui répondent en donnant un programme de production qui comporte des volumes pour chaque type de produit sur chaque journée sur un horizon de *M+2* à *M+12*. Tous les mois, ces volumes sont mis à jour pour prendre en compte les nouvelles données prévisionnelles.

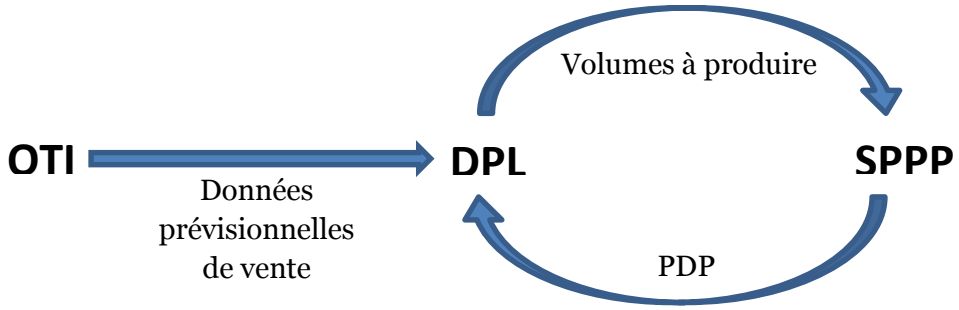


Figure 3.2 : Processus de planification du cas industriel

Un premier lissage est fait durant cette étape (par SPPP Bourg) qui consiste à étaler les volumes demandés par DPL pour chaque type de produit sur les différentes journées. Ce lissage est peu efficace puisque les volumes vont être modifiés à cause du processus de positionnement des commandes que nous détaillons dans la prochaine section. En effet, l'entreprise fabrique les véhicules industriels sur commande ce qui engendre un écart entre les prévisions et les commandes réelles des clients. Quand on s'approche du jour de production, on s'aperçoit que les volumes et les ratios des types de produit présentent des variations importantes entre les différentes journées d'une même semaine. Par conséquent, d'autres lissages s'imposent entre le PDP et le jour de production.

3.2.2 Processus de positionnement de commandes

Quand l'entreprise reçoit des commandes des clients, le service commercial (OTI) procède au positionnement de ces commandes sur les journées de production en tenant compte des volumes ouverts par le service planification de l'usine de Bourg en Bresse (lors du PIC/PDP) et des limitations des nombres de variantes par jour. Les limitations en termes de variantes sont appelées CIA (Contraintes Industrielles et d'Approvisionnement). Ce sont des limites capacitaires par jour qui modélisent les contraintes d'approvisionnement, les contraintes techniques et les contraintes d'organisation (humaines). Ces contraintes des lignes de production sont transmises sous forme de CIA à OTI pour les respecter lors du positionnement des commandes. Les niveaux des CIA sont revus tous les mois lors d'une réunion du SPPP avec le service méthode et OTI.

Le processus de positionnement des commandes continue jusqu'au jour du séquencement. Il est présenté dans la figure 3.3.

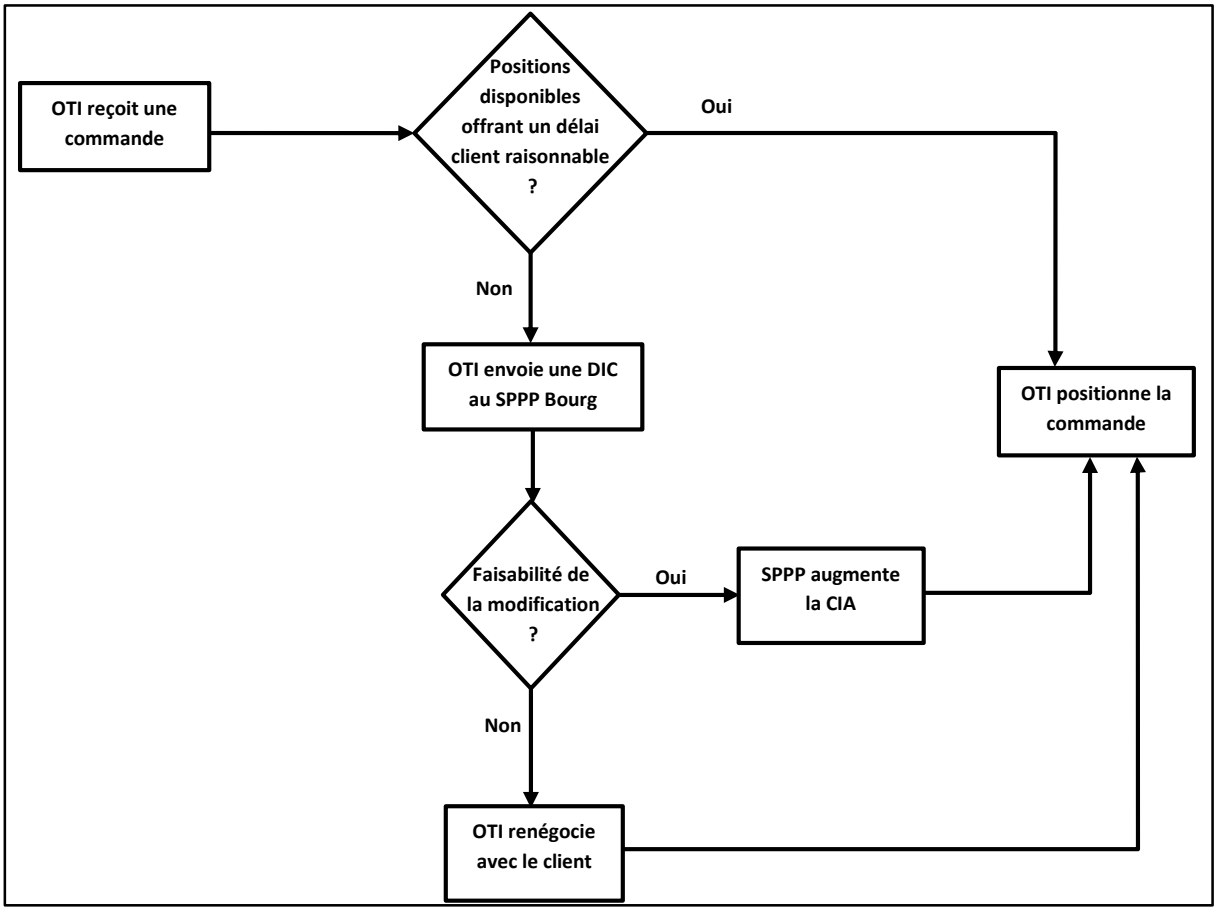


Figure 3.3 : Processus de positionnement des commandes

L'objectif du positionnement des commandes est de combler les trous sans excéder les volumes ouverts par SPPP. Par contre, pour ne pas provoquer une longue attente de certains clients, le service commercial peut demander d'ouvrir plus de volumes pour un

type de produit donné pour y positionner les commandes des clients correspondants : c'est ce qu'on appelle une Demande d'Inspection Commerciale (DIC). Le service de planification de l'usine de Bourg en Bresse étudie avec le service des méthodes de la ligne la faisabilité des modifications et ajoute des volumes (ou pas) pour le type de produit indiqué en diminuant les volumes des types de produit les moins consommés durant la période correspondant à la DIC, pour que le volume de fabrication par jour reste constant.

3.2.3 Lissage à la semaine

Nous remarquons un écart entre les données prévisionnelles et les commandes réelles dû aux changements que les clients engendrent à leurs commandes et les commandes urgentes reçues par le service commercial. Ceci déséquilibre le lissage fait lors du PIC/PDP. Le ratio des produits difficiles à fabriquer sur la ligne de production diffère d'une journée à l'autre. Ceci engendrait un déséquilibre de besoin de personnels sur les lignes de production d'une journée à l'autre. Mais, on ne peut pas embaucher des opérateurs pour une seule journée, et donc un nouveau lissage s'impose. C'est le lissage à la semaine. Il est effectué en deux étapes : lissage des silhouettes et lissage des variantes.

3.2.3.1 Lissage des silhouettes

La première étape du lissage à la semaine est le lissage des silhouettes. La silhouette est définie par le type et le nombre des ponts et essieux, et a été jugée comme le critère qui génère le plus de variabilité sur la ligne de production et par conséquent le critère qui fait l'objet du lissage de la production à la semaine.

Le lissage des silhouettes se fait une semaine avant qu'on ne commence à séquencer la première journée d'une semaine donnée. Ceci correspond à 16 à 21 jours avant le jour de fabrication. La figure 3.4 présente une partie du lissage de la semaine 46. Les différents jours d'une même semaine doivent avoir le même ratio de produits « lourds » (8x4, TRM et 6x2), le même ratio de produits « moyens » (6x4) et le même ratio de produits « légers » (4x2). Rappelons que ces regroupements de produits (lourds, moyens et légers) sont faits à ce stade sur la base des silhouettes des véhicules industriels.

Le lissage à la semaine est fait à l'aide d'un outil développé en interne. Il permet d'avoir une production lissée d'un jour à l'autre d'une même semaine mais aussi de faire un pré-séquencement à la journée. Le fonctionnement de l'outil se fait de la manière suivante : tout d'abord, on positionne les produits « lourds » (les plus difficiles à produire) en les espaçant le plus possible. Ce sont les produits en rouge dans la figure 3.4. Dans l'exemple de la figure 3.4, on privilégie les espacements de trois puis de deux entre les produits « lourds ». Ensuite, on positionne les produits « moyens » où il y a le plus

d'espacements entre les produits « lourds ». Ce sont les produits en jaune dans la figure 3.4. Enfin, on comble les trous avec les produits « légers » (produits en vert dans la figure 3.4). Le résultat est une production lissée avec des valeurs proches des ratios des silhouettes des produits sur tous les jours de la semaine.

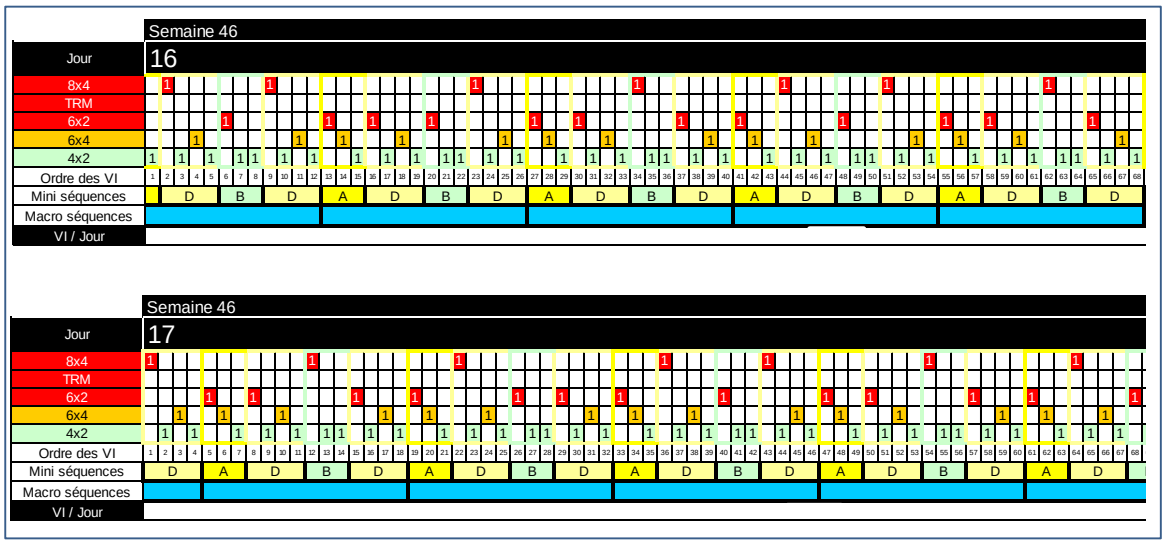


Figure 3.4 : Lissage à la semaine (exemple de résultat pour deux jours d'une même semaine)

Le lissage à la semaine permet aussi de communiquer une séquence fixe aux lignes de production (cette séquence n'est fixe que par rapport au critère des silhouettes des véhicules industriels). Cette séquence d'un certain nombre de produits se répétera presque toute la semaine. Son but est d'avoir une répétitivité et une visibilité facilitant le travail sur les lignes de production (ligne « macro séquence » dans la figure 3.4).

La figure 3.5 indique les résultats du lissage des silhouettes pour la semaine 46. Les produits « lourds » sont bien répartis sur tous les jours de la semaine.

Pour obtenir ce lissage, SPPP est obligé de demander à OTI de déplacer quelques commandes d'un jour à l'autre d'une même semaine. Ceci n'affecte pas le délai de livraison pour le client puisque ce délai s'exprime en semaines et non pas en jours.

Après le lissage des silhouettes, OTI s'engage, dans le cas de déplacement ou d'annulation d'une commande après ce lissage à la semaine, de la remplacer par une autre commande de même silhouette afin de préserver le résultat du lissage.

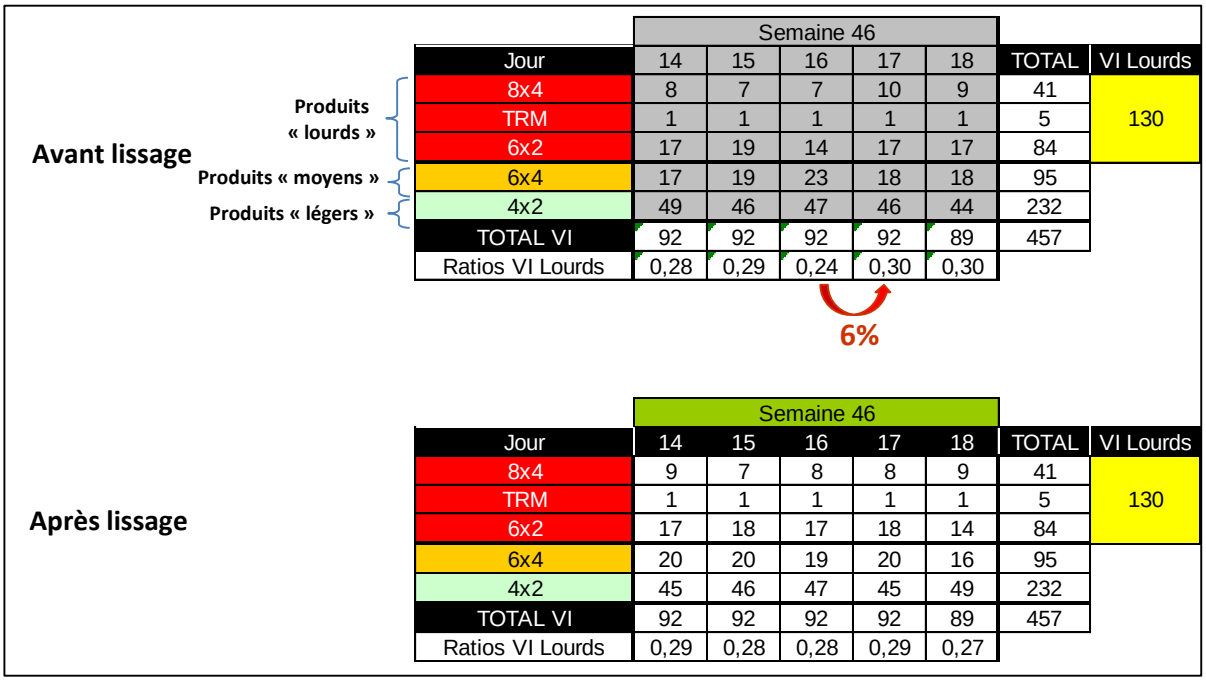


Figure 3.5 : Effet du lissage à la semaine

3.2.3.2 Lissage des variantes

Le lissage des variantes est effectué à la suite du lissage des silhouettes (pendant la même journée). Il permet de lisser sur tous les jours de la semaine les variantes les plus contraignantes dans la ligne de production sans toucher au lissage des silhouettes. Après consultation du service des méthodes de production, SPPP a choisi quatre variantes à lisser par ligne d'assemblage. Ce sont les variantes qui induisent le plus de variabilité des temps opératoires. La figure 3.6 montre un exemple illustrant la différence entre la répartition des variantes avant et après le lissage.

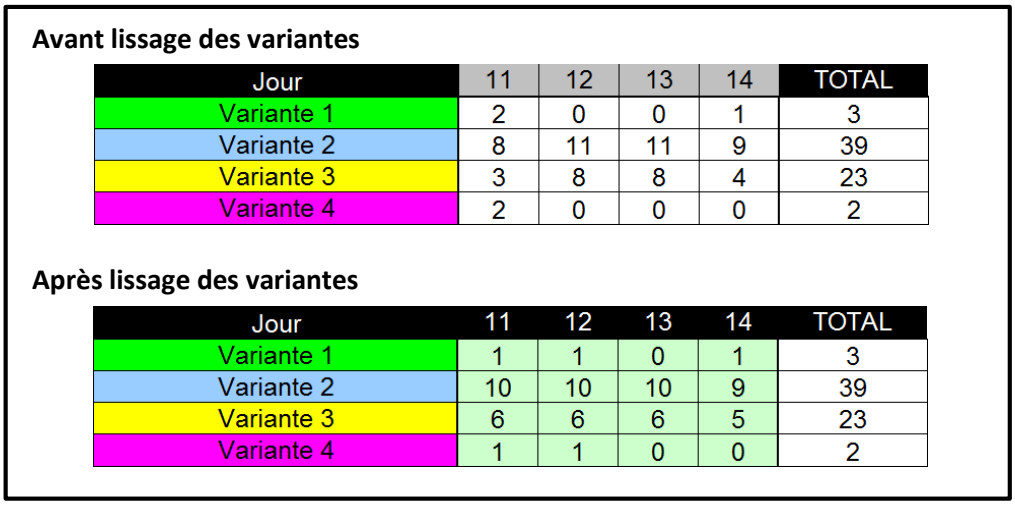


Figure 3.6 : Effet du lissage des variantes

Notons qu'OTI ne s'engage pas (comme dans le cas du lissage des silhouettes) à remplacer une commande annulée ou modifiée par une autre avec les mêmes variantes puisqu'il est difficile de trouver une commande avec la même silhouette et les mêmes variantes. Donc le lissage des variantes n'est pas un lissage ferme et la répartition des produits peut être modifiée jusqu'au jour du séquencement, mais les modifications des commandes ne sont pas fréquentes et le lissage reste efficace pour ces quatre variantes.

3.2.4 Le séquencement

Quinze jours avant la fabrication de la commande (ce nombre de jours peut varier selon les besoins de planification), on procède au lancement : la commande passe par un processus de validation et devient une commande ferme. On associe un numéro de VI (Véhicule Industriel) au numéro de commande.

Juste après le lancement (processus fait le matin du J-15 par OTI), le service planification de l'usine de Bourg en Bresse procède au séquencement (l'après-midi du même jour). Le séquencement définit l'ordre de passage des véhicules sur la ligne d'assemblage pour la journée de production concernée. L'approche utilisée est celle du *car sequencing* (chapitre 2, section 2.5). Le séquencement est fait à l'aide d'un outil s'appelant « séquence V6 » (utilisé à l'usine de Bourg en Bresse depuis Avril 2008). Cet outil utilise le tri glouton et le recuit simulé pour trouver la solution au problème de séquencement en respectant des règles : contraintes d'espacement et de succession. Un ordre de priorité de respect est donné aux différentes règles de séquencement. Dans le cas général, les règles de séquencement représentent des contraintes d'approvisionnement de la ligne ou des contraintes de capacité des ressources (charge des opérateurs de la ligne). Dans notre cas d'étude, les règles de séquencement reflètent des contraintes de capacité des ressources.

La figure 3.7 montre l'outil « Séquence v6 ». Les lignes représentent la liste des véhicules et les colonnes représentent la liste des règles à respecter. Quand un véhicule a l'option régie par une règle, le chiffre « 1 » apparait à l'intersection.

Le processus du séquencement présente quelques limites. En effet, avant l'introduction du lissage des silhouettes à la semaine dans le processus de planification, on utilisait l'outil « Séquence V6 » qui séquençait automatiquement les produits. Cependant, cet outil ne prend pas en compte la séquence fixe retenue lors du lissage : l'outil ordonnance les produits sur une journée et « casse » la séquence fixe. La séquence fixe obtenue au lissage à la semaine a été jugée importante et il faut la respecter lors du séquencement. On est donc obligé de séquencer manuellement. « Séquence v6 » est donc actuellement utilisé comme un outil de visualisation et non pas d'optimisation. Il permet à l'utilisateur de changer les positions des produits et de visualiser la séquence obtenue. Ceci demande

plus de temps pour la tâche de séquencement et introduit une source d'erreur sur les non respects des règles. Ordonnancer plus de 100 véhicules par jour en respectant plus de 30 règles est une tâche difficile surtout que les règles sont parfois complexes.

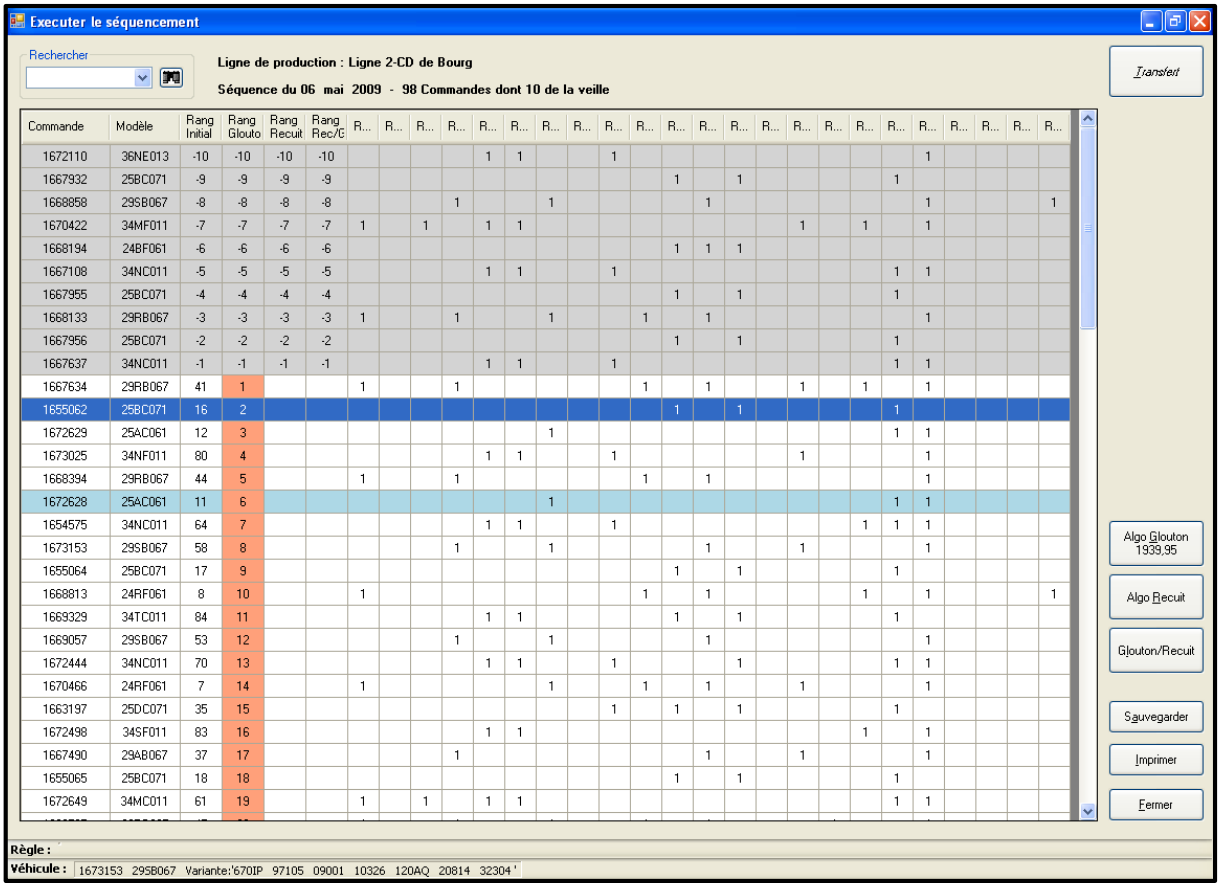


Figure 3.7 : Outil de séquencement « Séquence V6 »

La deuxième limite du séquencement dans notre cas d'étude est que les règles à respecter sont définies empiriquement. Les règles ne sont pas basées sur des études de temps opératoires pour déterminer les bons ratios à respecter. Le séquencement est basé sur des contraintes d'espacement et de succession empiriques basées sur les observations des managers des unités élémentaires de production (UEP) suite à des enchainements de produits jugés difficiles pour les opérateurs. Il est impossible donc de prévoir si de nouveaux enchainements poseront des problèmes sur les lignes et de connaître les nouvelles règles lors de l'introduction de nouveaux produits.

La troisième limite est que l'outil ne permet pas de vérifier facilement la qualité du lissage. En effet, si on ne respecte pas une règle, l'outil n'affiche pas ce non-respect alors qu'il peut causer de graves conséquences sur le fonctionnement de la ligne de montage.

L'analyse précédente des outils actuellement à la disposition de SPPP amènent une question principale. L'agrégation des données en utilisant les règles de séquencement constitue une perte d'information, n'est-t-il pas mieux d'intégrer tous les temps

d'opération et séquencer sur cette base pour optimiser le lissage de la production ? Dans la deuxième partie de cette thèse, nous nous attacherons à répondre à cette question. Avant de les aborder, nous décrivons rapidement le processus actuel d'équilibrage de charge dont les résultats sont une entrée à notre problème de séquencement.

3.3 L'équilibrage de charge

L'équilibrage des charges de travail est fait par le service méthode à chaque changement de cadence des lignes de production. Pour ceci, les équilibreurs utilisent un outil de visualisation et d'aide à l'équilibrage appelé ELGA. Dans cet outil, on trouve la liste précise des opérations pour chaque véhicule. Chaque opération doit être effectuée sur certains types de produits auxquels on attribue des taux d'apparition calculés sur la base des ratios actuels et des prévisions de ventes. En affectant des opérations à un opérateur pour un temps de cycle donné, l'outil nous propose une visualisation du taux de charge de cet opérateur. L'outil ne fait pas l'équilibrage en automatique. Il est fait manuellement par les équilibreurs à l'aide de fichiers Excel. La connaissance du graphe du montage (règles de succession entre les opérations) et de la disposition des moyens techniques est nécessaire à l'équilibrage, informations qui ne sont pas présents dans ELGA. ELGA permet seulement d'affecter des opérations à des opérateurs et de visualiser le résultat.

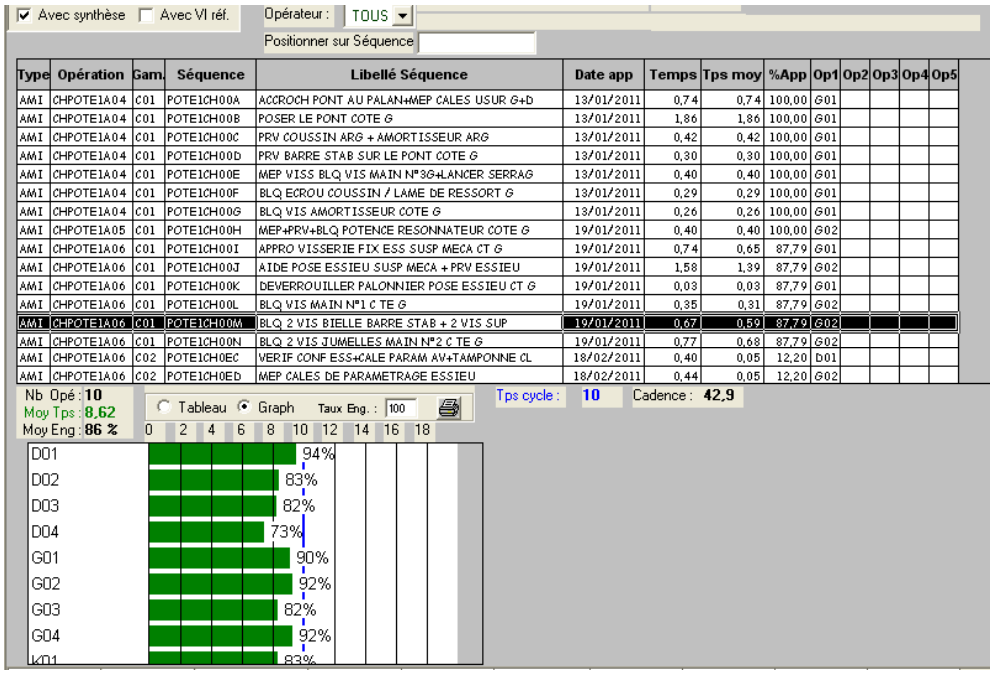


Figure 3.8 : Visualisation des taux de charges sur l'outil ELGA

Précisons que pour l'usine de Bourg en Bresse trois types d'opérateurs sont définis et l'équilibrage de charge affecte des opérations à ces différents types en tenant bien sûr compte des spécificités de chaque type que nous allons préciser maintenant :

- Les opérateurs type 1 : Ce sont les opérateurs usuels. Ils effectuent des opérations sur tous les véhicules du film de production. Le temps moyen disponible par véhicule pour ces opérateurs est le temps de cycle.
- Les opérateurs type 2 : Les tâches affectées à ces opérateurs sont à effectuer sur certains véhicules seulement (véhicules avec certaines variantes). Les opérateurs type 2 travaillent pendant un nombre connu de temps de cycle. Ce nombre diffère d'un véhicule à l'autre selon les variantes. Le temps disponible pour ces opérateurs dépend alors des véhicules.
- Les opérateurs type 3 : Pendant l'équilibrage de charge, certaines opérations ayant des temps opératoires relativement longs et présentes sur tous les véhicules pourraient conduire à un mauvais équilibrage puisqu'elles forcent le temps de cycle à être inefficacement grand induisant des temps d'inactivité pour d'autres opérateurs, d'où la solution d'avoir plusieurs opérateurs travaillant alternativement (opérateurs type 3). Leurs opérations doivent être effectuées sur tous les véhicules de la séquence et requièrent un nombre connu de temps de temps de cycle. Ce nombre est le même pour tous les véhicules. Des opérateurs de type 3 travaillent en alternant les véhicules entre eux selon leurs positions dans la séquence. Par exemple, si on a un temps disponible de deux temps de cycle, deux opérateurs type 3 vont travailler alternativement. Un opérateur travaillera sur les véhicules pairs et l'autre sur les véhicules impairs.

Il y a une distinction entre un opérateur et un poste de travail. Un poste de travail peut comporter plusieurs opérateurs. Dans l'exemple de la figure 3.9, on a 7 postes de travail et 10 opérateurs. Les opérateurs 1 à 3, 6 et 8 à 10 sont de type 1. Ils ont chacun un temps de cycle comme temps moyen disponible. L'opérateur 7 est un opérateur de type 2. Son temps disponible dépend des véhicules (jusqu'à trois temps de cycle). Les opérateurs 4 et 5 sont deux opérateurs type 3 qui alternent les véhicules et chacun d'eux a deux temps de cycle pour effectuer ces tâches. Les opérateurs d'un même poste de travail (ou d'une même zone), notamment les opérateurs de type 2 et 3, travaillent de façon indépendante.

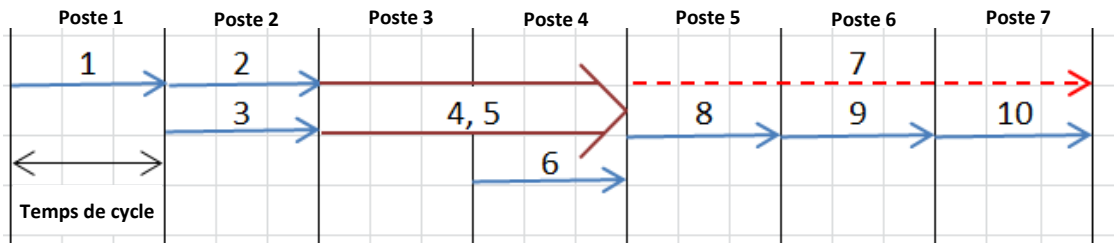


Figure 3.9 : Exemple d'une configuration d'opérateurs

3.4 Conclusion

Le cadre d'étude de la thèse se décrit essentiellement par :

- Le séquencement actuel utilise l'approche du « *car sequencing* » et se fait manuellement.
- Les règles du séquencement sont définies empiriquement.
- L'équilibrage de charge se fait manuellement.
- L'équilibrage de charge considère trois types d'opérateurs.

L'objectif de ce chapitre était de décrire fidèlement le cadre d'étude pratique de la thèse, qui servira de base pour les contributions de recherche présentées dans les chapitres suivants.

Deuxième partie : Résolution du problème de séquencement

Chapitre 4 : Choix du critère d’optimisation

4.1 Introduction

Ce chapitre a pour objet la description du choix du critère d'optimisation. Ce choix est une étape cruciale dans les problèmes d'optimisation industriels puisque le critère doit refléter les situations réelles du problème et les objectifs souhaités par l'industriel. Afin de choisir le critère d'optimisation, nous expliquons d'abord la problématique du séquencement du cas d'étude. Cette analyse révèle des situations critiques auxquelles les opérateurs sont confrontés et nous permet d'orienter le choix du critère. Nous finissons par donner les notations des données et des variables du problème à résoudre.

4.2 La problématique

Comme précisé dans les chapitres précédents, l'approche du « *car sequencing* » utilise des règles de séquencement (des contraintes d'espacement et de succession) qui permettent de modéliser les surcharges des opérateurs. La diversité des produits engendre que les règles de séquencement sont de plus en plus difficiles à définir puisqu'elles deviennent des combinaisons de plusieurs variantes. Lorsque les contraintes sont mal définies, elles ne parviennent plus à saisir la vraie surcharge de travail à minimiser.

Nous rappelons que lorsque la diversité des produits est importante, il devient donc judicieux de supprimer les règles de séquencement (celles qui reflètent des contraintes de charge des opérateurs) et de les remplacer par le lissage des charges de travail réelles à réaliser par chaque opérateur afin de supprimer le risque des règles mal définies et de saisir la vraie surcharge de travail à minimiser. Dans notre cas d'étude, toutes les règles de séquencement reflètent des contraintes de charge des opérateurs et non pas des contraintes de disponibilité des pièces : remplacer l'approche du « *car sequencing* » par l'utilisation directe des temps opératoire (c'est-à-dire l'approche du « *mixed-model sequencing* ») est donc possible.

Dans notre cas d'étude, les postes de travail sont fermés (cf. chapitre 2, section 2.6.2) et donc les opérateurs n'ont pas la possibilité de travailler en dehors de leurs zones définies pour des raisons de non disponibilité des outils spécifiques d'assemblage. Les opérateurs sont donc obligés de finir leurs tâches dans leurs zones de travail. La zone de travail est équivalente à un temps de cycle pour les opérateurs de type 1 et à un multiple de temps de cycle pour les opérateurs de types 2 et 3 (voir chapitre 3, section 3.3). Cependant, lors de l'équilibrage de charge et à des fins d'optimisation, des opérations ayant des temps opératoires théoriques supérieurs au temps de cycle (ou au temps disponible pour les opérateurs de types 2 et 3) sont attribuées aux opérateurs. Pour ces

opérations, les opérateurs sont donc obligés de se précipiter pour finir leurs tâches avant la fin du temps correspondant à leurs zones de travail afin de ne pas arrêter la ligne. Les opérateurs doivent donc accélérer la réalisation de leurs tâches pour finir avant que le véhicule n'atteigne la limite de la zone de travail, ce qui engendre de la fatigue et du stress. Quand ces véhicules se succèdent, ceci conduit les opérateurs à se mettre en mauvaises postures, d'une part, et à occasionner des défauts qualité à cause des oublis dus à la précipitation d'autre part. Dans des cas rares, afin d'éviter l'arrêt de la ligne, un opérateur polyvalent peut aider l'opérateur en difficulté à finir ces tâches.

Comme toutes les opérations dans un poste donné doivent être terminées avant la sortie du poste, les opérateurs du poste suivant peuvent toujours commencer leurs tâches dès que les véhicules sont disponibles dans leurs zones de travail. Cette dernière observation de notre cas d'étude permet de considérer comme indépendants les postes de travail.

4.3 Le critère d'optimisation

L'objectif de notre étude est de diminuer les situations critiques (stress, fatigue, mauvaises postures, défauts qualité). Nous allons donc prendre en compte les temps opératoires théoriques définis lors de l'équilibrage de charge.

On compte un retard (ou une surcharge) dès que le temps théorique des opérations sur un véhicule donné excède le temps correspondant à la zone de travail de l'opérateur. Le but est de minimiser les retards afin de diminuer les situations critiques liées à la succession de véhicules entrainant des retards.

Nous considérons tout d'abord une première approche qui suppose que l'opérateur commence les tâches sur tous les véhicules au début de son poste ce qui est théoriquement possible puisque tous les retards sont compensés. La figure 4.1 représentent deux séquences différentes des mêmes véhicules en appliquant cette première approche. Cependant, et comme le montre la figure 4.1, l'ordre des véhicules séquencés n'engendre aucune modification au critère de minimisation des retards. On a la même valeur du critère pour les deux séquences (la somme des retards c'est-à-dire les parties rayées des temps opératoires dans la figure 4.1), alors que, selon les agents de maitrise des lignes de montage, la séquence 1 est plus contraignante que la séquence 2 : l'opérateur est obligé dans la séquence 1 d'accélérer ses opérations pour deux véhicules de suite (véhicules A et B), ce qui engendre les situations critiques citées ci-dessus. Dans la séquence 2, le véhicule D permet à l'opérateur de récupérer suite à la surcharge entrainée par le véhicule A.

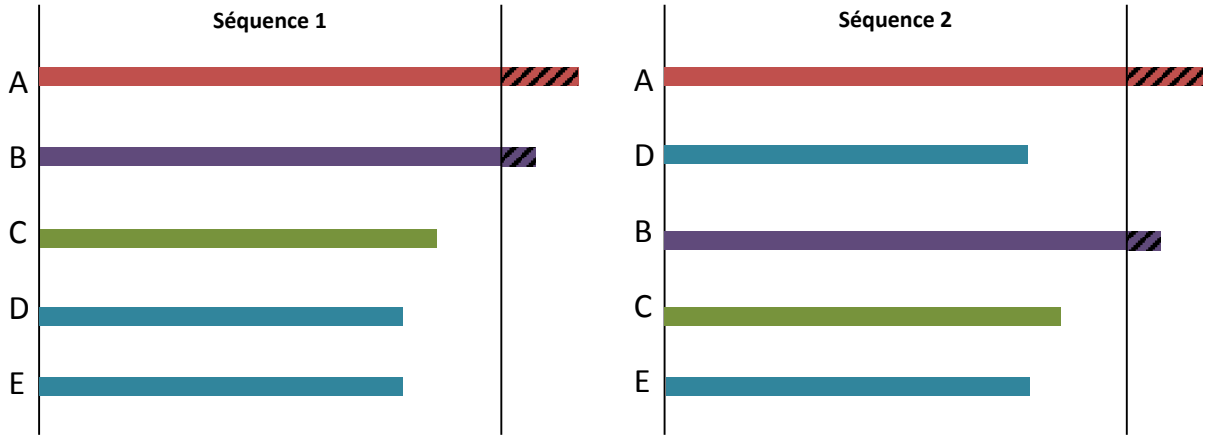


Figure 4.1 : Première approche de modélisation

La première approche est donc inappropriée aux attentes en termes d'objectif de la procédure du séquencement, il faut par conséquent trouver une autre modélisation qui permet d'éviter l'élément déclencheur des situations critiques qui est la succession des véhicules entrainant des retards.

La deuxième approche proposée suppose qu'un retard est répercuté sur le véhicule suivant dans la séquence : nous modélisons les retards qu'aurait l'opérateur s'il ne se précipitait pas et nous considérons que si un véhicule engendre un retard (même s'il est théorique), il se répercutera sur le véhicule suivant dans la séquence puisque l'opérateur sera fatigué du fait de la surcharge ; la vitesse d'exécution des tâches sur le véhicule suivant en sera diminuée. La minimisation des retards cumulés permet de sanctionner la succession de ces véhicules même si ces retards ne sont pas explicites puisque les surcharges sont compensées.

La figure 4.2 schématise la deuxième approche de modélisation sur les séquences déjà présentées dans la figure 4.1. À cause du retard subi sur le véhicule A, l'opérateur ne peut pas commencer le véhicule suivant dès son entrée dans la zone du travail. Dans la séquence 1, le retard sera cumulé avec celui du véhicule B. Dans la séquence 2, le retard est rattrapé grâce au véhicule D qui a un temps opératoire inférieur au temps de cycle. De plus, comme précédemment, l'opérateur ne peut pas prendre de l'avance et doit attendre l'entrée du véhicule dans sa zone de travail pour pouvoir effectuer ses tâches. On parle donc de repos (cas du véhicule E dans la séquence 1 et des véhicules B et E dans la séquence 2). Grâce à cette modélisation, la séquence 1, très contraignante pour l'opérateur dans la réalité, est pénalisée : la succession des véhicules « difficiles » est sanctionnée en répercutant le retard d'un véhicule sur le suivant dans la séquence.

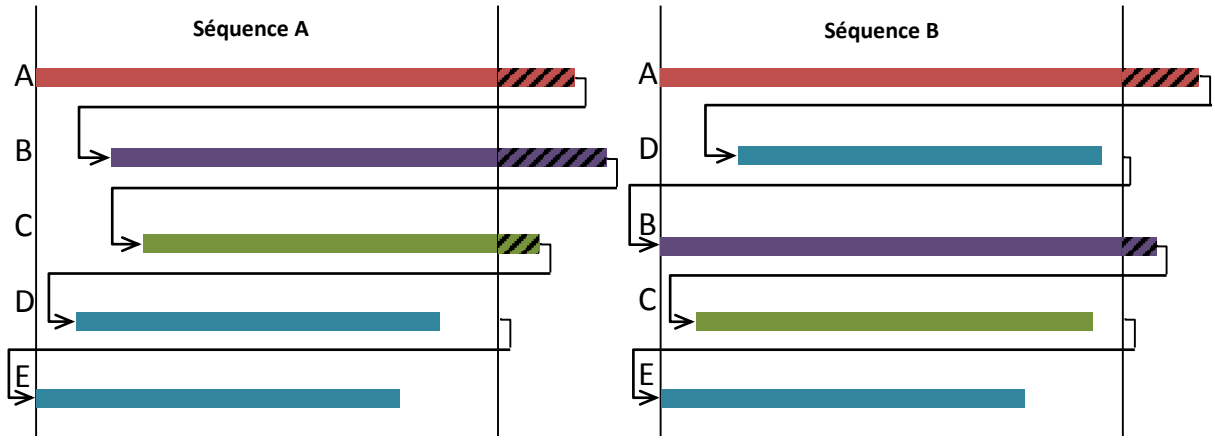


Figure 4.2 : Approche de modélisation retenue

Comme précisé ci-dessus, les retards sont compensés et n'ont pas de conséquence pour le poste suivant. La modélisation considère donc que les postes sont indépendants : les retards d'un poste n'influent pas sur le début des opérations dans le poste suivant.

Notre critère d'optimisation sera donc la minimisation des retards cumulés et ceci pour tous les opérateurs. Ce critère permet de minimiser les situations critiques. Cette modélisation a été validée par les ergonomes et les responsables de la ligne de montage de notre cas d'étude.

Cette modélisation est illustrée à l'aide d'un exemple pour chaque type d'opérateurs. Le tableau 4.1 présente les temps opératoires d'un exemple d'un opérateur de type 1 devant réaliser quatre produits p1, p2, p3 et p4. Le temps de cycle étant de 5 unités de temps.

Tableau 4.1 : Exemple d'un opérateur de type 1

Produit	**p1**	**p2**	**p3**	**p4**
Opérateur de type 1	5	6	3	4

La figure 4.3 est un diagramme de Gantt de la séquence p2, p1, p3 puis p4. On remarque que le retard subit par l'opérateur pour le véhicule p2 impacte le début des opérations sur le véhicule p1 qui génère aussi un retard à cause du cumul des retards. Quand l'opérateur finit ses tâches sur le véhicule p3, il ne peut commencer ses opérations sur le véhicule p4 que quand p4 est disponible dans son poste de travail : l'opérateur a une période de repos d'une unité de temps après le véhicule p3.

L'objectif est de trouver une séquence de véhicules avec le moins de retards cumulés possible (carreaux grisés dans la figure 4.3). Nous rappelons que ces retards ne sont pas réels. En réalité, les opérateurs se précipitent pour finir leurs opérations à temps. Ce sont des retards issus des temps opératoires théoriques définis lors de l'équilibrage des

charges, mais les minimiser permettra d'éviter les situations critiques dues à la précipitation.

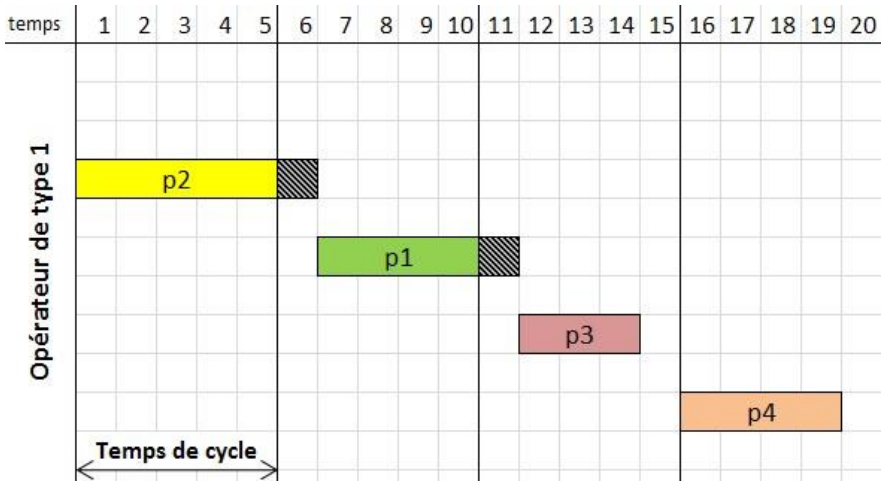


Figure 4.3 : Diagramme de Gantt d'un exemple d'un opérateur de type 1

La figure 4.4 montre le diagramme de Gantt d'un exemple d'un opérateur de type 2 sur une séquence de 8 véhicules, le temps de cycle étant de 3 unités de temps. L'opérateur n'a pas de tâches à effectuer sur les véhicules p2, p3, p5, p7 et p8, ils n'apparaissent donc pas dans le diagramme de Gantt. Pour les produits p1, p4 et p6 les temps opératoires sont de 10, 6 et 7 respectivement. Le nombre de pas est indiqué entre parenthèses pour chaque véhicule. Il correspond au nombre de temps de cycle disponibles pour l'opérateur pour un véhicule donné. Par exemple, la zone de travail de l'opérateur pour p6 correspond à trois temps de cycle. Les retards cumulés apparaissent quand l'opérateur atteint la limite de cette zone. Ils sont représentés par les carreaux grisés dans la figure 4.4.

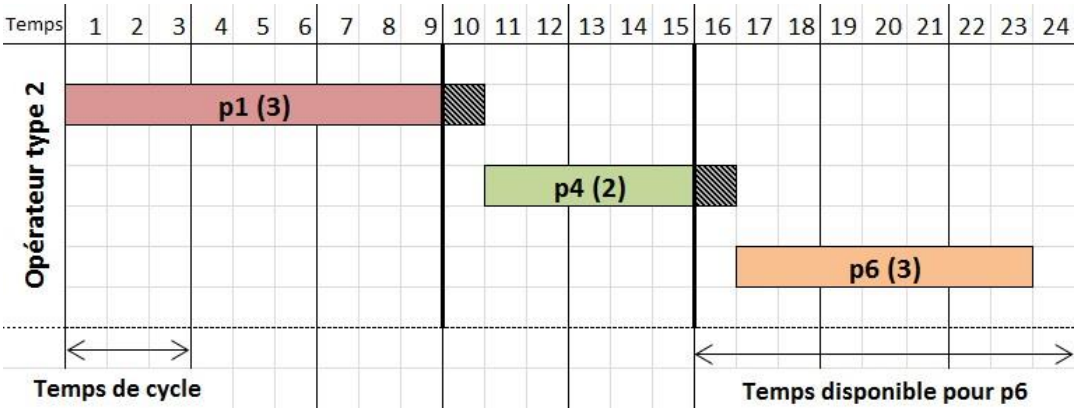


Figure 4.4 : Diagramme de Gantt d'un exemple d'opérateurs de type 2

La figure 4.5 illustre le diagramme de Gantt d'un exemple de trois opérateurs du troisième type travaillant en parallèle sur une séquence de sept véhicules. Les temps opératoires sur ces sept véhicules sont respectivement 10, 8, 9, 10, 8 et 7. Le temps

disponible pour chaque opérateur est alors de trois temps de cycle. Contrairement aux opérateurs type 2 le temps disponible ne dépend pas des véhicules. L'opérateur 1 effectue ses tâches sur les véhicules p1, p4 et p7. Les véhicules p2 et p5 sont traités par l'opérateur 2. Finalement l'opérateur 3 travaille sur les véhicules p3 et p6. Les retards cumulés à minimiser sont illustrés par les carreaux grisés dans la figure 4.5.

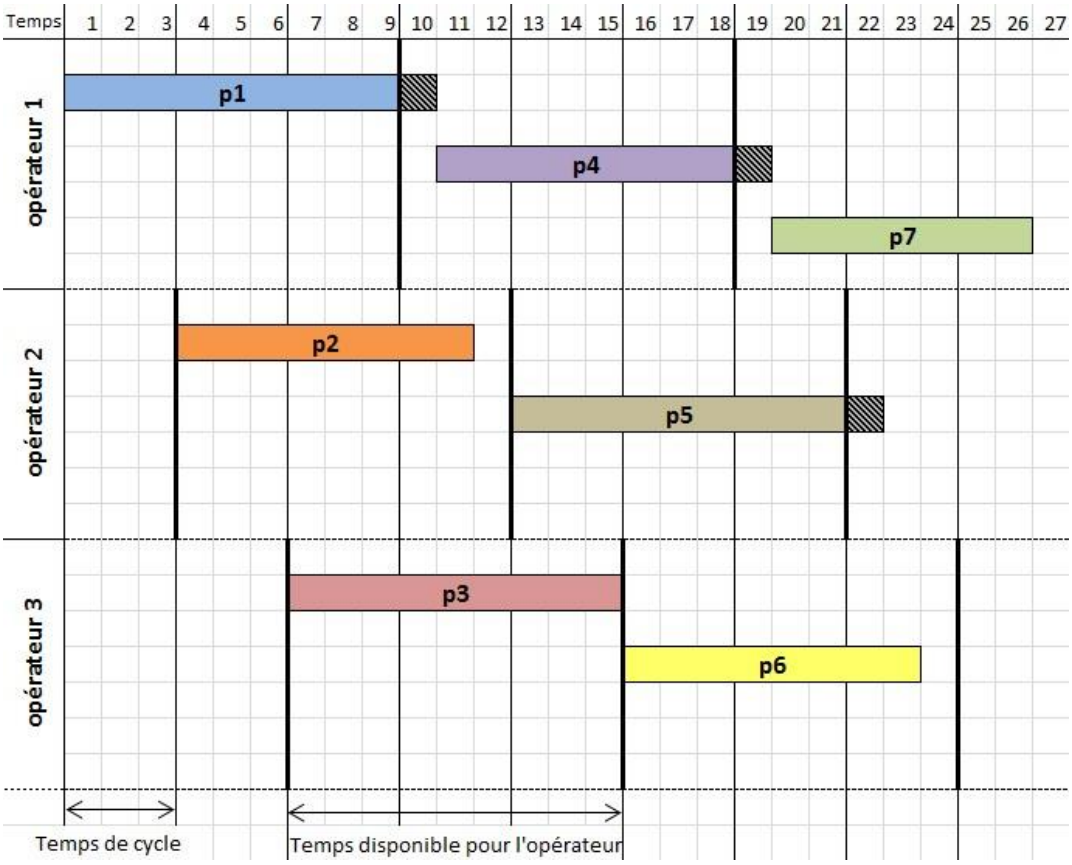


Figure 4.5 : Diagramme de Gantt d'un exemple d'opérateurs de type 3

Après avoir présenté la modélisation des trois types d'opérateurs, considérons maintenant le cas de la succession de deux opérateurs. Pour ceci, le tableau 4.2 présente les temps opératoires d'un exemple de deux opérateurs successifs de type 1 devant réaliser trois produits p1, p2 et p3. Le temps de cycle étant de cinq unités de temps. Cet exemple permet d'illustrer l'indépendance entre les opérateurs. Cette indépendance est aussi valable pour des opérateurs de types différents.

Tableau 4.2 : Exemple de deux opérateurs de type 1

Produit	**p1**	**p2**	**p3**
Opérateur 1	5	6	3
Opérateur 2	6	4	4

La figure 4.6 est un diagramme de Gantt de la séquence p2, p1 puis p3. Tout d'abord, la séquence est bien sûr la même pour les deux opérateurs. On remarque que le retard

observé pour l'opérateur 1 sur p2 n'impacte pas le début des opérations de l'opérateur 2 sur le même véhicule. Les retards modélisés sont théoriques puisque les surcharges sont compensées. Nous les prenons en compte afin de pénaliser la succession des véhicules engendrant des surcharges pour l'opérateur concerné. Mais quand un opérateur se précipite pour finir ses opérations sur un véhicule, cela aura un impact sur ses propres performances pour son travail sur le véhicule suivant dans la séquence, mais cela n'impactera pas les performances des opérateurs suivants dans la ligne.

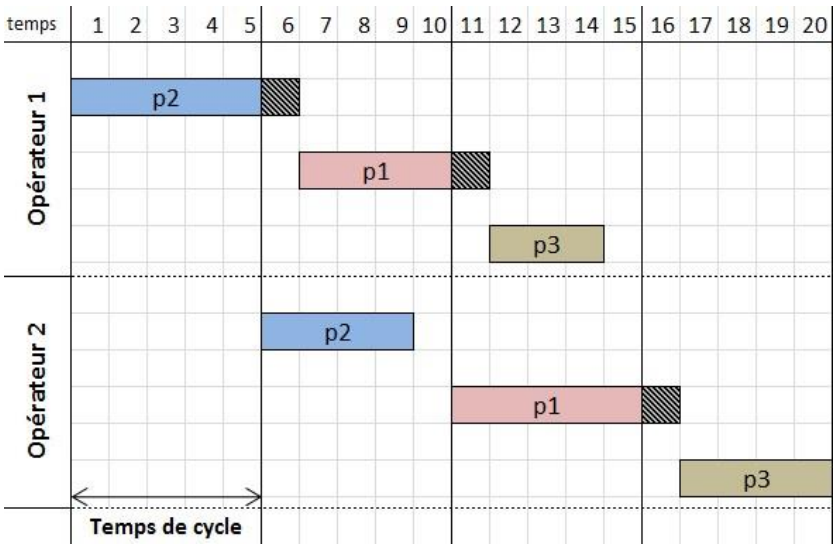


Figure 4.6 : Diagramme de Gantt d'un exemple d'opérateurs de type 1

4.4 Notations

Pour terminer, précisons les données et paramètres suivants qui seront utilisés dans les chapitres suivants pour la définition des modèles d'optimisation pour le problème à résoudre :

N	Nombre de produits
M	Nombre de modèles
P	Nombre d'opérateurs
P_1	Nombre d'opérateurs de type 1
P_2	Nombre d'opérateurs de type 2
P_3	Nombre d'opérateurs de type 3
i	Indice sur les produits
m	Indice des modèles
p	Indice des opérateurs
j	Indice des positions
N_m	Nombre de produits du modèle m
$\propto_{ip}^{2}$	Nombre binaire qui est égale à 1 si l'opérateur p de type 2 doit effectuer des opérations sur le produit i et 0 sinon

β_{ip}^2 Nombre de temps de cycle pendant lesquels l'opérateur p de type 2 peut travailler sur le produit i

$\propto_{jp}^3$ Paramètre binaire qui est égale à 1 si l'opérateur p de type 3 doit effectuer des opérations sur le produit de la position j et 0 sinon

β_p^3 Nombre de temps de cycle pendant lesquels l'opérateur p de type 3 peut travailler sur un produit (ce nombre est le même pour tous les produits)

γ Temps de cycle

t_{ip} Temps opératoire du produit *i* pour l'opérateur p

d_{ip} Retard ou repos pour le produit i pour l'opérateur p de type 1 et 2 : $d_{ip} = t_{ip} - \gamma$
Pour les opérateurs de type 3 : $d_{ip} = t_{ip} - \beta_p^3 * \gamma$

A Grand entier

Nous utiliserons les variables suivantes pour l'optimisation du problème à résoudre:

x_{ij} 1 si le produit i est affecté à la position j, 0 sinon

c_{jp} Retard ou repos cumulé de l'opérateur p pour le produit de la position j

r_{jp} Variable intermédiaire pour le calcul du retard cumulé de l'opérateur p à la fin de la réalisation du produit en position j. C'est plus précisément le retard cumulé qu'aurait l'opérateur p à la fin du produit j, si cet opérateur avait pour chaque produit un temps de cycle pour réaliser les opérations, et ceci quel que soit la durée des opérations (en particulier même pour les produits qui ne sont pas traités par l'opérateur).

b_{jp} Variable intermédiaire pour le calcul du retard de l'opérateur p de type 2 pour le produit de la position j. Ceci représente le temps dont dispose l'opérateur, en plus du temps de cycle, pour effectuer ses tâches (cette variable n'est pas significative pour le produits qui ne sont pas traités par l'opérateur p).

w_{jp} Retard cumulé de l'opérateur p pour le produit de la position j

4.5 Conclusion

Dans ce chapitre, nous avons justifié le choix du critère d'optimisation considéré dans notre problème. Il s'agit de la minimisation de la somme des cumuls des retards. Nous avons illustré ce critère par des exemples étudiant les trois types d'opérateurs étudiés. Dans les trois prochains chapitres, on abordera la résolution du problème du séquencement en utilisant ce critère d'optimisation. Des méthodes exactes et approchées seront utilisées.

Chapitre 5 : Résolution du problème de séquencement par la programmation linéaire mixte

5.1 Introduction

Nous avons défini dans le chapitre précédent la problématique et le critère d'optimisation issus des observations des situations rencontrées dans notre cas d'étude. Nous entamons la résolution du problème de séquencement avec minimisation des retards cumulés. Les rares instances de tailles réelles présentées dans la littérature sont issues de l'industrie automobile et résolues en utilisant des heuristiques. Dans notre cas d'étude, les volumes de production journaliers sont moins importants que dans l'industrie automobile : les instances sont donc de tailles plus réduites. De plus, l'indépendance des opérateurs (une hypothèse issue du cas d'étude) apporte une simplification au problème. Il est donc intéressant d'étudier les performances de méthodes exactes sur les instances réelles de notre cas d'étude. Le problème de séquencement des véhicules en prenant en compte les temps opératoires est prouvé NP-difficile même pour le cas mono-opérateur (Kotani * et al., 2004; Tsai, 1995; Yano et Rachamadugu, 1991).

Dans ce chapitre, nous proposons une modélisation du problème considéré utilisant la programmation linéaire mixte. Cette modélisation permet de résoudre le problème mais aussi de le formaliser. La formulation proposée est d'abord testée sur des instances académiques. Nous procédons ensuite à une étude expérimentale de la complexité. Enfin, nous testons le programme linéaire sur des instances issues de notre cas d'étude dont les résultats seront analysés pour discerner l'apport et les limites de la procédure proposée. Le travail exposé dans ce chapitre a donné lieu aux publications suivantes : (Aroui et al., 2013, 2014a).

5.2 Formulation

La formulation du problème est une étape cruciale dans un problème d'optimisation. Nous proposons un modèle mono-opérateur pour avoir un modèle de base qu'on complexifiera ensuite en intégrant l'aspect multi-opérateurs.

5.2.1 Formulation mono-opérateur

Nous décrivons la modélisation du problème du séquencement sur une ligne d'assemblage multi-modèles pour la minimisation des retards cumulés. Dans cette section, on considère que la ligne est composée d'un seul opérateur et que cet opérateur est de type 1. Pour illustrer le problème monoposte, prenons l'exemple du séquencement de cinq produits. Les temps opératoires de chaque produit sont donnés dans le tableau 5.1, le temps de cycle étant de 5 unités de temps.

Tableau 5.1 : Temps opératoires de l'exemple

Produit	***p1***	***p2***	***p3***	***p4***	***p5***
Temps	5	6	4	3	3

La figure 5.1 montre le diagramme de Gantt de la production des produits dans l'ordre p4, p2, p1, p5 puis p3. Nous pouvons remarquer que le retard provoqué par p2 n'est pas considéré comme une opération inachevée, mais répercuté sur le début des opérations pour le produit p1 qui finit donc aussi en retard. Pour commencer les opérations sur le produit p3, l'opérateur ne peut pas aller au-delà de la limite de début de poste et doit attendre que le produit soit disponible dans sa zone : un temps d'inactivité (repos) se produit. Notons qu'une des séquences optimales (avec une seule unité de retard) est obtenue en échangeant p1 et p5.

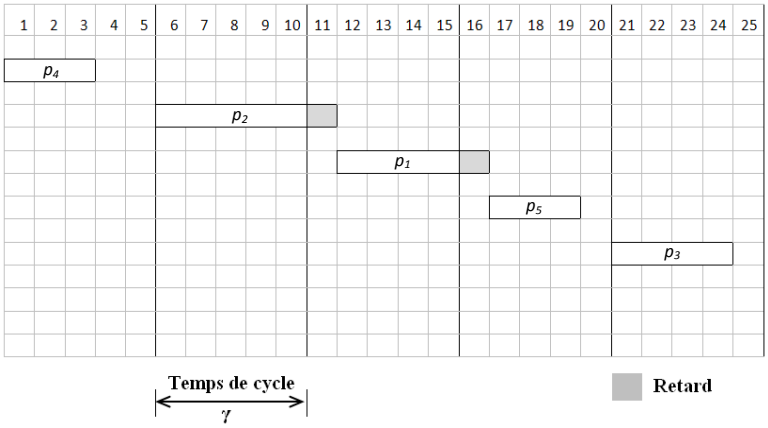


Figure 5.1 : Diagramme de Gantt de l'exemple

Nous adaptons les paramètres et les variables données dans le chapitre 4 pour obtenir les notations suivantes (on supprime l'indice p de l'opérateur puisqu'on ne modélise qu'un seul opérateur) :

Données :

t_i Temps opératoire du produit i

d_i Retard (si $d_i > 0$) ou repos (si $d_i \leq 0$) pour le produit i : $d_i = t_i - \gamma$
(le retard cumulé ou le repos du produit i dans la séquence dépendra de d_i et des produits antécédents dans la séquence)

Variables:

x_{ij} 1 si le produit i est affecté à la position j, 0 sinon

c_j Retard ou repos cumulé à la position j

w_j Retard cumulé à la position j : $w_j = \max(0, c_j)$

Le tableau 5.2 présente les valeurs de variables c_j et w_j pour la séquence p4, p2, p1, p5 puis p3 de l'exemple donné dans le tableau 5.1 et illustré par le diagramme de Gantt de la figure 5.1. Notre modèle cherche à minimiser les retards cumulés de l'opérateur et non pas les repos. C'est pour cette raison que w_j ne prend que les valeurs positives de c_j.

Tableau 5.2 : Valeurs des variables de l'exemple

Variables	$\sum_{i=1}^{n} d_i x_{ij}$	c_j	w_j
p4	-2	-2	0
p2	1	1	1
p1	0	1	1
p5	-2	-1	0
p3	-1	-1	0

La formulation du problème de séquencement sur une ligne d'assemblage multi-modèles pour la minimisation des retards pour un opérateur de type 1 est donc :

$$\min f = \sum_{j=1}^{N} w_j$$

Sous contraintes:

$$c_j = w_{j-1} + \sum_{i=1}^{N} d_i x_{ij} \qquad \forall j \in [1..\mathrm{N}] \qquad (1)$$

$$w_j \geq 0 \qquad \forall j \in [1..\mathrm{N}] \qquad (2)$$

$$w_j \geq c_j \qquad \forall j \in [1..\mathrm{N}] \qquad (3)$$

$$\sum_{j=1}^{N} x_{ij} = 1 \qquad \forall i \in [1..\mathrm{N}] \qquad (4)$$

$$\sum_{i=1}^{N} x_{ij} = 1 \qquad \forall j \in [1..\mathrm{N}] \qquad (5)$$

$$x_{ij} \in \{0,1\} \qquad \forall i \in [1..\mathrm{N}], \forall j \in [1..\mathrm{N}] \qquad (6)$$

$$c_j \in \mathbb{R}, w_j \in \mathbb{R}^+, w_0 = 0 \qquad \forall j \in [1..\mathrm{N}] \qquad (7)$$

La fonction objectif minimise la somme des retards cumulés pour l'opérateur. Les contraintes (1) calculent le retard ou le repos sur le poste de travail pour les produits pour

toute position j. Les contraintes (2) et (3) linéarisent l'expression des variables w_j. Les contraintes (4) et (5) assurent respectivement l'affectation de chaque produit à une seule position et de chaque position à un seul produit.

Le modèle mono-opérateur n'est pas pertinent dans la réalité puisque la ligne d'assemblage est constituée de plusieurs opérateurs mais il permet d'avoir un modèle de base. On prendra en compte l'aspect multi-opérateurs dans la prochaine section.

5.2.2 Formulation multi-opérateurs

Dans cette section, nous présentons la modélisation du problème du séquencement sur une ligne d'assemblage multi-modèles avec plusieurs opérateurs pour la minimisation des retards. Comme la ligne est tractée, l'ordre de passage des produits est le même pour tous les opérateurs. Nous considérons les trois types d'opérateurs issus de notre cas d'étude décrit dans le chapitre 3. Les notations utilisées pour formuler le problème sont données dans le chapitre 4 (section 4.4).

Ce modèle a pour objectif de minimiser la somme des retards cumulés pour tous les opérateurs de la ligne d'assemblage. La différence entre w_{jp} et r_{jp} est que w_{jp} est le retard cumulé de l'opérateur p et que r_{jp} est le retard cumulé de l'opérateur p s'il n'a qu'un temps de cycle pour finir ses opérations sur le produit j. Les opérateurs de type 1 ont un temps de cycle comme temps disponible, leurs retards w_{jp} sont égaux à r_{jp}. Pour les opérateurs de type 2 et 3, r_{jp} sert à calculer les retards cumulés w_{jp}.

La formulation du problème de séquencement sur une ligne d'assemblage multi-modèles pour la minimisation des retards pour le cas multi-opérateurs est :

$$\min f = \sum_{p=1}^{P} \sum_{j=1}^{N} w_{jp}$$

Sous contraintes:

$$c_{jp} = r_{j-1,p} + \sum_{i=1}^{N} d_{ip} x_{ij} \qquad \forall j \in [1..\mathrm{N}], \forall p \in [1..\mathrm{P}] \qquad (8)$$

$$r_{jp} \geq 0 \qquad \forall j \in [1..\mathrm{N}], \forall p \in [1..\mathrm{P}] \qquad (9)$$

$$r_{jp} \geq c_{jp} \qquad \forall j \in [1..\mathrm{N}], \forall p \in [1..\mathrm{P}] \qquad (10)$$

-- Retards des opérateurs de type 1 --

$$w_{jp} = r_{jp} \qquad \forall j \in [1..\mathrm{N}], \forall p \in [1..P_1] \qquad (11)$$

-- Retards des opérateurs de type 2 --

$$b_{jp} = \gamma \left(\left(\sum_{i=1}^{N} \beta_{ip}^{2}\, x_{ij} \right) - 1 \right) \qquad \forall j \in [1..\mathrm{N}], \forall p \in [1..P_2] \qquad (12)$$

$$w_{jp} \geq 0 \qquad \forall j \in [1..\mathrm{N}], \forall p \in [1..P_2] \qquad (13)$$

$$w_{jp} \geq r_{jp} - b_{jp} - A \left(1 - \sum_{i=1}^{N} \alpha_{ip}^{2} x_{ij} \right) \qquad \forall j \in [1..\mathrm{N}], \forall p \in [1..P_2] \qquad (14)$$

-- Retards des opérateurs de type 3 --

$$w_{jp} \geq 0 \qquad \forall j \in [1..\mathrm{N}], \forall p \in [1..P_3] \qquad (15)$$

$$w_{jp} \geq \alpha_{jp}^{3} \left(w_{j-\beta_{p}^{3},p} + \sum_{i=1}^{N} x_{ij} d_{ip} \right) \qquad \forall j \in [1..\mathrm{N}], \forall p \in [1..P_3] \qquad (16)$$

$$\sum_{j=1}^{N} x_{ij} = 1 \qquad \forall i \in [1..\mathrm{N}] \qquad (17)$$

$$\sum_{i=1}^{N} x_{ij} = 1 \qquad \forall j \in [1..\mathrm{N}] \qquad (18)$$

$$x_{ij} \in \{0,1\} \qquad \forall i \in [1..\mathrm{N}], \forall j \in [1..\mathrm{N}] \qquad (19)$$

$$c_{jp}, r_{jp}, b_{jp}, w_{jp} \in \mathbb{R}, r_{0p} = 0 \qquad \forall j \in [1..\mathrm{N}], \forall p \in [1..\mathrm{P}] \qquad (20)$$

Les contraintes (8) à (10) sont similaires aux contraintes (1) à (3) du modèle mono-opérateur. d_{ip} représente un repos si sa valeur est négative et représente un retard sinon. Ceci correspond au repos ou le retard qu'aurait l'opérateur p si le produit i passe seul dans la ligne de montage. Si le produit i est attribué à la position j de la séquence, son retard dépendra de d_{ip} et du retard cumulé qu'aurait l'opérateur p à la fin du produit j-1, si l'opérateur avait pour chaque produit un temps de cycle, représenté par $r_{j-1,p}$. Les contraintes (8) assurent que le retard ou le repos cumulé à la position j de l'opérateur p est son retard cumulé à la position $j-1$ s'il avait pour chaque produit un temps de cycle en plus du retard (ou du repos) du produit séquencé à la position j. Puisqu'on ne prend en

compte que les retards dans la fonction objectif (et non pas les repos), les contraintes (9) et (10) indiquent que $r_{jp} = \max(0, c_{jp})$.

Pour les opérateurs de type 1, le temps disponible est un temps de cycle donc leurs retards sont égaux à la variable intermédiaire r_{jp} (contraintes 11).

Les contraintes (12)-(14) calculent les retards des opérateurs du type 2. Si l'opérateur p n'a pas d'opérations à effectuer sur le produit de la position j, on ne compte pas de retard ($w_{jp} = 0$). Le retard de l'opérateur p de type 2 pour le produit de la position j est :

$$w_{jp} = \begin{cases} \max\left(r_{jp} - b_{jp}\ ,\ \ 0\right) & \text{si } \sum_{i=1}^{N} \alpha_{ip}^{2} x_{ij} = 1 \\ 0 & \text{sinon} \end{cases}$$

Avec $b_{jp} = \gamma\left(\sum_{i=1}^{N} \beta_{ip}^{2}\ x_{ij} - 1\right)$

Cette équation est illustrée par les contraintes (12)-(14). b_{jp} représente le temps dont dispose l'opérateur, en plus du temps de cycle, pour effectuer ses tâches. Si un opérateur a un temps disponible de trois temps de cycle pour effectuer ses tâches sur un produit, b_{jp} sera égal à deux fois le temps de cycle. Comme r_{jp} est calculé en prenant en compte que le temps disponible est égal à un seul temps de cycle, ceci sera corrigé en soustrayant b_{jp} pour les opérateurs de type 2. La partie $\{\ A\left(1 - \sum_{i=1}^{N} \alpha_{ip}^{2}\ x_{ij}\right)\}$ de la contrainte (14) permet de ne pas compter de retard pour des produits que l'opérateur ne traite pas. Notons qu'on calcule b_{jp} pour tous les produits et donc aussi pour les produits non traités mais que ça valeur n'est alors pas utilisée

Les contraintes (15) et (16) calculent les retards des opérateurs de type 3. Le retard qu'accumule l'opérateur p de type 3 pour le produit de la position j est :

$$w_{jp} = \begin{cases} \max\left(0, w_{j-\beta_p^3,p} + \sum_{i=1}^{N} d_{ip} x_{ij}\right) & \text{si } \alpha_{jp}^{3} = 1 \\ 0 & \text{sinon} \end{cases}$$

C'est le retard ou repos brut (d_{ip}) du produit de la position j ajouté au retard cumulé du produit précédent traité par l'opérateur p (si l'opérateur a des opérations à effectuer sur le produit de la position j, le produit précédent est situé à la position $j - \beta_p^3$). On ne compte pas de retard pour des produits que l'opérateur ne traite pas ($\alpha_{jp}^{3} = 0$).

Les contraintes (17) à (19) sont similaires aux contraintes (4) à (6) du modèle mono-opérateur. Les contraintes (17) garantissent qu'une seule position est attribuée à chaque

produit et les contraintes (18) assurent qu'un seul produit est attribué à chaque position de la séquence. Les contraintes (19) indiquent que les variables affectées sont binaires.

Le tableau 5.3 donne les valeurs des données et des variables pour l'exemple des opérateurs de type 1 présenté dans le chapitre 4. Les deux opérateurs sont indépendants donc le retard observé pour l'opérateur 1 sur *p2* n'impacte pas la valeur du retard de l'opérateur 2 sur le même véhicule.

Tableau 5.3 : Données et variables pour l'exemple des opérateurs de type 1

		Donnée	*Variables*	
Positions	*Séquence*	d_j	c_j	w_j
1	*p2*	1	1	1
2	*p1*	0	1	1
3	*p3*	-2	-1	0
4	*p4*	-1	-1	0

Le tableau 5.4 donne les valeurs des données et des variables pour l'exemple des opérateurs de type 2 avec un temps de cycle égal à 3 présenté dans le chapitre 4. L'opérateur n'a pas de tâches à effectuer sur les véhicules *p2*, *p3*, *p5*, *p7* et *p8* donc aucun retard n'est comptabilisé pour ces véhicules.

Tableau 5.4 : Données et variables pour l'exemple des opérateurs de type 2

	Données			*Variables*		
Séquence	t_{ip}	β_{ip}^2	α_{ip}^2	r_{jp}	b_{jp}	w_{jp}
p1	10	3	1	7	6	1
p2	0	0	0	4	-3	0
p3	0	0	0	1	-3	0
p4	6	2	1	4	3	1
p5	0	0	0	1	-3	0
p6	7	3	1	5	6	0
p7	0	0	0	2	-3	0
p8	0	0	0	0	-3	0

Le tableau 5.5 donne les valeurs des données et des variables pour le premier opérateur de l'exemple des opérateurs de type 3 présenté dans le chapitre 4 (figure 4.5). L'opérateur 1 effectue ses tâches sur les véhicules *p1*, *p4* et *p7*. Aucun retard n'est

comptabilisé pour les autres véhicules. Si le produit *p2* était séquencé en première position, l'opérateur 1 aurait un temps opératoire de 8 unités de temps à cette position.

Tableau 5.5 : Données et variables pour l'exemple des opérateurs de type 3

	Données			*Variable*
Séquence	t_{i1}	α_{j1}^{3}	d_{i1}	w_{j1}
p1	10	1	1	1
p2	8	0	-1	0
p3	9	0	0	0
p4	9	1	0	1
p5	10	0	1	0
p6	8	0	-1	0
p7	7	1	-2	0

Dans les prochaines sections, le programme linéaire est expérimenté pour observer ses limites : nous testons tout d'abord le modèle mono-opérateur puis le modèle multi-opérateurs. Pour les deux modèles, on procède à une analyse numérique de complexité et des tests sur les données industrielles. Des instances académiques seront aussi testées sur le modèle multi-opérateurs.

5.3 Cas mono-opérateur

Les résultats des études expérimentales de ce chapitre sont obtenus par le solveur IBM ILOG CPLEX Optimization Studio V12.4 tournant sur un PC Dell avec Intel Core i5 2.50 GHz et 4GB de RAM utilisant Windows 7 Enterprise.

5.3.1 Analyse numérique de la complexité dans le cas mono-opérateur

Dans cette partie, nous procédons à une analyse numérique de la complexité mais avant cela nous montrons la difficulté de cerner expérimentalement les facteurs de complexité du programme linéaire. Dans le tableau 5.6, nous présentons deux instances avec 20 produits et 12 modèles. La seule différence entre ces deux instances est que le 15ème produit engendre plus de repos (on passe de d_{15}=-1 à d_{15}=-1,8). Les deux instances sont donc très proches : nous nous attendons donc à des temps de résolution assez proches. Les tests montrent le contraire. La solution optimale de la première instance a

été obtenue au bout de 20 secondes alors que celle de la deuxième a été obtenue au bout de 145 secondes. L'analyse de la complexité d'un point de vue expérimental est donc difficile à faire puisque des instances très proches peuvent nécessiter des temps de calcul très différents pour être résolues.

Tableau 5.6 : Différence des temps opératoires pour des instances similaires

Première instance (20 sec)				Deuxième instance (145 sec)			
Produit *i*	d_i	Produit *i*	d_i	Produit *i*	d_i	Produit *i*	d_i
1	-1,8	11	0,8	1	-1,8	11	0,8
2	1	12	0,4	2	1	12	0,4
3	-0,8	13	0,8	3	-0,8	13	0,8
4	-0,2	14	0	4	-0,2	14	0
5	0,4	15	-1	5	0,4	15	-1,8
6	-1,2	16	-1,6	6	-1,2	16	-1,6
7	0,8	17	-1,4	7	0,8	17	-1,4
8	-1	18	-0,4	8	-1	18	-0,4
9	1	19	0,4	9	1	19	0,4
10	-1,6	20	0	10	-1,6	20	0

Malgré la difficulté de l'analyse numérique de complexité, nous cherchons à trouver les effets du nombre de produits et du nombre de modèles sur le temps de calcul du programme linéaire mono-opérateur (de type 1).

Nous générons 20 instances pour chaque couple (nombre de produits, segment de nombre de modèles). Nous arrêtons l'optimisation au bout de 10 minutes de calcul. Notons que la solution optimale peut avoir été trouvée par CPLEX sans pouvoir prouver son optimalité en moins de 10 minutes. Le tableau 5.7 présente les résultats de cette analyse. Chaque cellule indique le nombre d'instances (sur les 20 instances générées) pour lesquelles la solution optimale a été obtenue et prouvée en moins de 10 minutes. Par exemple, pour 30 produits et pour un nombre de modèles qui varie entre 8 et 11, la solution optimale a été obtenue pour 8 instances sur les 20 générées en moins de 10 minutes.

Nous constatons que le nombre de tests pour lesquels la solution optimale est obtenue et prouvée en moins de 10 minutes augmente lorsque le nombre de produits et le nombre de modèles diminuent. Ceci montre expérimentalement que le nombre de modèles et le nombre de produits sont des facteurs de complexité pour le modèle mono-opérateur.

Tableau 5.7 : Analyse numérique de complexité du cas mono-opérateur

Nombre de produits / **Segment de nombre de modèles**	**20 produits**	**30 produits**	**40 produits**	**50 produits**	**60 produits**
18-29 modèles		2/20	0/20	0/20	0/20
12-16 modèles	19/20	2/20	0/20	0/20	1/20
8-11 modèles	20/20	8/20	4/20	1/20	0/20
5-6 modèles	20/20	18/20	10/20	2/20	3/20

5.3.2 Tests sur des données industrielles

Pour constituer des instances pour le cas mono-opérateur à partir des donnés du cas d'étude, nous prenons les temps opératoires d'un seul opérateur parmi tous les opérateurs de la ligne d'assemblage où la cadence est de 43 camions par équipe. Six instances, correspondant à six opérateurs avec différents taux de charges, ont été testées pour une journée de production donnée (tableau 5.8).

Tableau 5.8 : Tests sur les données industrielles

Instances	Taux de charge	Nombre de produits	Temps de calcul (sec)
1	82,3%	43	228,0
2	79%	43	1,3
3	91,6%	43	0,5
4	90,3%	43	3,0
5	97,7%	43	0,9
6	69,1%	43	19,0

La procédure utilisée actuellement (respect de contraintes d'espacement et de succession) prend en compte des contraintes liées à plusieurs postes de travail de la ligne d'assemblage. On ne peut pas donc faire une comparaison avec les résultats de la procédure actuelle. Néanmoins, ces tests nous permettent de valider le modèle mono-opérateur : toutes les instances ont été résolues en moins de 19 secondes sauf la première qui a été résolue en moins de 4 minutes.

5.3.3 Propriétés de la solution optimale

L'observation des résultats nous a permis de trouver quelques caractéristiques de la solution optimale pour le cas mono-opérateur (de type 1).

Toute solution au problème est une succession de sous-séquences de produits dont les temps opératoires sont inférieurs au temps de cycle et de sous-séquences de produits dont les temps opératoires sont supérieurs au temps de cycle. Nous donnons ci-après deux propriétés que nous avons observées dans la solution optimale :

Propriété 1 : Dans une sous-séquence de produits ayant des temps opératoires supérieurs au temps de cycle, ces produits sont classés dans l'ordre croissant de leurs temps opératoires (voir démonstration ci-dessous).

Propriété 2 : Dans une sous-séquence de produits ayant des temps opératoires inférieurs au temps de cycle, ces produits sont classés dans l'ordre croissant de leurs temps opératoires (voir démonstration dans l'annexe A).

Démonstration de la propriété 1

Démontrons que dans la solution optimale, toute sous-séquence de produits ayant des temps opératoires supérieurs au temps de cycle est classée par ordre croissant des temps opératoires.

Supposons qu'on ait deux produits *a* et *b* à séquencer en *j* et *j*+1 tels que $t_a > \gamma$ et $t_b > \gamma$. Notons w_{j-1} le retard cumulé jusqu'à la position *j*-1.

Les retards des produits *a* et *b* sont:

$d_a = t_a - \gamma$ et $d_b = t_b - \gamma$

On considère le cas : $t_a \leq t_b$ et donc $d_a \leq d_b$. Deux cas se présentent selon les positions de *a* et de *b* :

1. Les produits *a* et *b* sont attribués respectivement aux positions *j* et *j*+1 c'est-à-dire dans l'ordre croissant des temps opératoires.

Calculons le retard cumulé à chaque position :

$w_j = \max(0 , c_j)$ avec $c_j = w_{j-1} + d_a$

Comme $w_{j-1} \geq 0$ et $d_a > 0$ alors $c_j \geq 0$

⇨ $w_j = c_j = w_{j-1} + d_a$

De même, $w_{j+1} = c_{j+1} = w_j + d_b$

Alors $w_{j+1} = w_{j-1} + d_a + d_b$

La somme des retards (fonction objectif) est alors :

$f(croissant) = w_j + w_{j+1} = 2.w_{j-1} + 2.d_a + d_b$

2. Les produits *a* et *b* sont attribués respectivement aux positions $j+1$ et j c'est-à-dire dans l'ordre décroissant des temps opératoires. On aboutit par un calcul similaire à :

$f(décroissant) = w_j + w_{j+1} = 2.w_{j-1} + 2.d_b + d_a$

Puisque $d_a \leq d_b$ on a $f(croissant) \leq f(décroissant)$.

Le classement dans la solution optimale est donc l'ordre croissant des temps opératoires.

La démonstration de la deuxième propriété est similaire à la première. Elle est présentée dans l'annexe A.

5.4 Cas multi-opérateurs

Dans cette section, nous allons nous intéresser au problème de séquencement en prenant en compte plusieurs opérateurs. Tout d'abord, nous procédons à des tests académiques pour valider notre modèle. Ensuite, nous analysons expérimentalement la complexité de notre modèle en nous basant sur les temps de calcul. Nous testons notre modèle sur des instances industrielles dans un deuxième temps.

5.4.1 Tests académiques

5.4.1.1 Instances issues de la littérature

Cette étude expérimentale est basée sur des instances issues de la littérature. Ce sont 225 instances issues de l'étude de (Bautista et Cano, 2008). Le modèle qu'ils proposent considère que les postes sont dépendants. En effet, le modèle donne aux opérateurs de chaque poste de travail la possibilité de continuer les opérations au-delà du temps de cycle ce qui influe sur le début des opérations des postes suivants. Pour cette raison, une comparaison de notre critère avec celui du modèle proposé par (Bautista et Cano, 2008) n'est pas significative. Le but de cette étude est de valider notre modèle avec des instances de référence issue de la littérature et plus précisément de vérifier s'il converge sur toutes ces instances. Les instances ont les caractéristiques suivantes :

- 16 produits, 4 modèles et 4 opérateurs de type 1.
- Chaque instance est construite en combinant un programme de production et une structure de temps opératoires.
- 45 programmes de production, c'est-à-dire 45 profils de demande, présentés dans le tableau 5.9. Les programmes de 1 à 4 sont des programmes avec une grande

demande d'un des quatre modèles ; les programmes de 5 à 10 considèrent deux modèles avec une grande demande ; les programmes de 11 à 17 sont des programmes équilibrés ; un des modèles a une faible demande dans les programmes de 18 à 21 ; et les programmes de 22 à 45 ont un des modèles avec une faible demande et un autre avec une forte demande.

- 5 structures de temps opératoires données dans le tableau 5.10 : la première structure présente des temps opératoires proche du temps de cycle ; des temps opératoires distants du temps de cycle sont donnés dans la deuxième structure ; la troisième structure a un déséquilibre entre les deux premiers et les deux derniers opérateurs ; deux modèles peuvent provoquer des retards à tous les opérateurs dans la quatrième structure ; et dans la cinquième structure chaque opérateur a des temps opératoires importants pour un modèle donné.
- Le temps de cycle est de 100 unités de temps.

Tableau 5.9 : Programmes de production des instances de référence

Programme		1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21			
Modèles	m_1	13	1	1	1	7	7	7	1	1	1	5	5	5	3	3	3	4	5	5	5	1			
	m_2	1	13	1	1	7	1	1	7	7	1	5	3	3	5	3	3	4	5	5	1	5			
	m_3	1	1	13	1	1	7	1	7	1	7	3	5	3	5	5	4	4	5	1	5	5			
	m_4	1	1	1	13	1	1	7	1	7	7	3	3	5	3	5	4	4	1	5	5	5			
Programme		22	23	24	25	26	27	28	29	30	31	32	33	34	35	36	37	38	39	40	41	42	43	44	45
Modèles	m_1	1	1	1	1	1	1	3	3	3	3	3	3	5	5	5	5	5	5	7	7	7	7	7	7
	m_2	3	3	5	5	7	7	1	1	5	5	7	7	1	1	3	3	7	7	1	1	3	3	5	5
	m_3	5	7	3	7	3	5	5	7	1	7	1	5	3	7	1	7	1	3	3	5	1	5	1	3
	m_4	7	5	7	3	5	3	7	5	7	1	5	1	7	3	7	1	3	1	5	3	5	1	3	1

Dans (Bautista et Cano, 2008), le retard est calculé par rapport à la limite du poste (différente du temps de cycle). Nous avons alors adapté notre modèle pour qu'il tienne compte de cette particularité en ne calculant les retards qu'à partir de cette limite. Dans le tableau 5.10, l_p représente cette limite pour l'opérateur p.

Les résultats sont présentés dans l'annexe B. On trouve les solutions optimales pour les 225 exemples en moins d'une seconde en moyenne. Le programme de production numéro 17 est le programme qui nécessite le plus de temps de calcul (9 secondes en moyenne pour toutes les structures de temps opératoires). Notre modèle peut donc résoudre sans problème des instances de référence issues de la littérature.

Tableau 5.10 : Temps opératoires des instances de référence

	Structure 1				Structure 2				Structure 3			
Opérateur / *Modèle*	1	2	3	4	1	2	3	4	1	2	3	4
m_1	92	103	101	95	91	120	90	100	111	120	85	82
m_2	97	98	105	104	80	105	113	107	114	113	100	94
m_3	103	104	99	96	107	88	117	86	83	85	115	119
m_4	108	95	95	105	114	87	100	114	98	87	110	115
l_p	108	105	106	106	115	120	120	115	115	120	115	120

	Structure 4				Structure 5			
Opérateur / *Modèle*	1	2	3	4	1	2	3	4
m_1	113	119	115	116	115	99	104	96
m_2	114	113	112	118	104	119	100	102
m_3	82	85	84	87	89	98	114	87
m_4	95	87	94	81	95	87	85	118
l_p	115	120	115	120	115	120	115	118

5.4.1.2 Instance aves les trois type d'opérateurs

La modélisation des opérateurs de type 2 et 3 constitue l'une des originalités de nos travaux. A notre connaissance, il n'existe pas d'instances issues de la littérature avec ces types d'opérateurs. Pour valider notre modèle, nous générons donc 3 instances incorporant ces types d'opérateurs. Le temps de cycle est de 7 minutes pour les trois instances.

La première instance comporte un opérateur de type 1, un opérateur de type 2 et trois opérateurs de type 3. Nous cherchons la séquence optimale de 6 produits. Les temps opératoires de cette instance sont donnés dans le tableau 5.11. Pour l' opérateur de type 2, nous mettons entre parenthèses le nombre de temps de cycle nécessaires à la réalisation des opérations sur le produit considéré. Trois opérateurs de type 3 alternent les produits de la ligne. Pour chacun d'eux, le temps disponible pour effectuer des opérations sur un produit est de trois temps de cycle. La solution optimale est trouvée en moins d'une seconde. La séquence optimale de l'instance 1 est {*p2*, *p5*, *p1*, *p6*, *p4*, *p3*} avec un retard cumulé de 6,02 minutes.

La deuxième instance comporte 2 opérateurs de chaque type et 6 produits. Les temps opératoires de cette instance sont donnés dans le tableau 5.12. Les deux opérateurs de type 3 alternent les produits de la ligne. Pour chacun d'eux, le temps disponible pour effectuer des opérations sur un produit est de deux temps de cycle.

Tableau 5.11 : Temps opératoires de l'instance académique 1

	Opérateurs type 1	*Opérateurs type 2*	*Opérateurs type 3*		
Opérateur / *Produit*	1	2	3	4	5
p1	8	15 (2)	22	22	22
p2	5,84	0	24	24	24
p3	5	7,2 (1)	21	21	21
p4	9	0	22	22	22
p5	8	13 (2)	20	20	20
p6	6,5	0	21	21	21

La solution optimale est trouvée en moins d'une seconde. La séquence optimale de l'instance 2 est {*p4, p3, p6, p5, p2, p1*}avec un retard cumulé de 10,02 minutes.

Tableau 5.12 : Temps opératoires de l'instance académique 2

	Opérateurs type 1		*Opérateurs type 2*		*Opérateurs type 3*	
Opérateur / *Produit*	1	2	3	4	5	6
p1	6,1	7,25	19,5 (3)	0	15	15
p2	7,84	5,98	14,4 (2)	21,2 (3)	14	14
p3	6,1	8,64	0	0	13,5	13,5
p4	8	5,9	0	7,3 (1)	12,5	12,5
p5	6,4	6,6	0	0	13,5	13,5
p6	6,4	6,8	0	5 (2)	14	14

La troisième instance est un exemple de 12 produits et 10 opérateurs (tableau 5.13) : 5 opérateurs de type 1, 2 opérateurs de type 2 et 3 opérateurs de type 3. Pour chacun des opérateurs de type 3, le temps disponible pour effectuer des opérations sur un produit est trois temps de cycle.

La solution optimale est obtenue en moins de 2 secondes. Une des séquences optimales de l'instance 3 est {*p12, p11, p9, p4, p6, p2, p7, p10, p3, p8, p5, p1*} avec un retard cumulé de 19,46 minutes.

Les tests académiques permettent ainsi de valider le modèle en programmation linéaire. Notre modèle peut résoudre sans problème des instances de référence issues de la littérature et des instances avec les trois types d'opérateurs.

Tableau 5.13 : Temps opératoires de l'instance académique 3

	Opérateurs type 1					*Opérateurs type 2*		*Opérateurs type 3*		
Opérateur / *Produit*	1	2	3	4	5	6	7	8	9	10
p1	7,24	8,59	6,74	5	8,8	20 (3)	13,5 (2)	22	22	22
p2	5,84	5,86	7,07	5,28	7,42	0	0	20	20	20
p3	5,84	5,86	5,53	5,28	4,74	14,3 (2)	19 (3)	21	21	21
p4	6,54	6,04	6,74	5	8,8	0	0	22	22	22
p5	5,84	6,19	6,82	5,49	4,74	0	0	19	19	19
p6	5,84	5,86	5,53	5,28	4,74	27,5 (4)	19 (3)	23	23	23
p7	7,24	6,97	5,75	7,29	4,74	0	0	20	20	20
p8	5,84	6,19	6,74	5	8,8	0	0	22	22	22
p9	5,84	5,86	6,41	5,28	4,74	0	0	19	19	19
p10	5,84	6,04	7,36	5	7,42	0	0	19	19	19
p11	7,24	8,74	6,74	5	8,8	0	13,5 (2)	21	21	21
p12	5,84	7,15	6,41	5,49	4,74	14,3 (2)	0	21	21	21

5.4.2 Analyse numérique de la complexité dans le cas multi-opérateurs

Dans cette partie nous allons analyser numériquement la complexité du modèle afin de déterminer ses limites et ses facteurs de complexité. Pour ceci, nous considérons un cas de base issu de notre cas d'étude avec 4 opérateurs de type 1, 20 produits et 5 modèles. Le cas de base est présenté dans le tableau 5.14 (retard ou repos par véhicule par poste). Nous rappelons que les éléments négatifs correspondent à des repos. La ligne « Intervalle » indique le minimum et le maximum des retards (ou repos) pour chaque opérateur.

Des premiers tests montrent que les paramètres susceptibles d'être des facteurs de complexité sont le nombre de produits et le taux de charge mais aussi le nombre de modèles et bien sûr le nombre d'opérateurs. Pour tester plus précisément l'effet de chaque paramètre, nous augmentons progressivement sa valeur en fixant les autres paramètres. Les données de ces tests sont générées aléatoirement à partir du cas de base (voir la procédure de génération des données en annexe C). Nous limitons le temps de calcul à 30 minutes.

Tableau 5.14 : Cas de base pour l'analyse numérique de la complexité

Modèles	Produits	Opérateur 1	Opérateur 2	Opérateur 3	Opérateur 4
m1	*p1* à *p3*	0,3	0,1	-2,2	-1,1
m2	*p4* à *p8*	0,1	1,3	-1,6	-0,7
m3	*p9* à *p13*	-0,5	0,3	-2,1	-1,1
m4	*p14* à *p17*	-1,1	-2,4	-3,1	1
m5	*p18* à *p20*	-1,1	-2,4	2	-1
Intervalle		[-1,1 ; 0,3]	[-2,4 ; 1,3]	[-3,1 ; 2]	[-1,1 ; 1]
Taux de charge		95,9%,	96,1%,	85,8%	94,9%

Dans le tableau 5.15, nous faisons varier le nombre d'opérateurs (de 4 à 24) : nous générons de nouveaux temps opératoires pour ajouter des opérateurs au cas de base (tableau 5.14) tout en gardant les autres paramètres constants (voir la procédure de génération des données en annexe C). Tous les opérateurs sont de type 1. Chaque ligne du tableau représente la moyenne du temps de calcul de quatre instances avec le même nombre d'opérateurs.

Tableau 5.15 : Effet du nombre des opérateurs à séquencer

Nombre d'opérateurs	Temps de calcul (sec)
4	463
8	419
12	466
16	*1501
20	*1682
24	*1800

Le signe (*) indique qu'au moins un des quatre tests ne trouve pas la solution optimale (ou n'arrive pas à prouver son optimalité) au bout de 30 minutes de calcul. Dans ce cas, la valeur donnée est une borne inférieure au temps de calcul puisque nous prenons pour ces instances un temps de calcul égal à 30 minutes (soit 1800 secondes). Dans le tableau 5.15, le temps de calcul croît lorsque le nombre d'opérateurs augmente. À partir de 16 opérateurs, certaines instances ne trouvent pas les solutions optimales ou n'arrivent pas à prouver l'optimalité des solutions trouvées en moins de 30 minutes.

Le tableau 5.16 présente l'évolution du temps de calcul en fonction du nombre de produits. Nous remarquons que le temps de calcul croît considérablement avec le nombre de produits. Nous notons qu'à partir de 24 produits le programme linéaire ne trouve pas

les solutions optimales (ou n'arrive pas à prouver leur optimalité) en moins de 30 minutes de calcul pour les instances testées.

Tableau 5.16 : Effet du nombre de produits à séquencer

Nombre de produits	Temps de calcul (sec)
20	463
22	*1139
24	*1800
26	*1800
28	*1800
30	*1800

Dans le tableau 5.17, nous testons l'effet du nombre de modèles sur le temps de calcul. Nous augmentons progressivement le nombre de modèles (jusqu'à 20 modèles) tout en gardant les autres paramètres fixes.

Tableau 5.17 : Effet du nombre de modèles

Nombre de modèles	Temps de calcul (sec)
5	463
10	*1088
15	*1800
20	*1800

Le temps de calcul augmente aussi lorsque le nombre de modèles augmente. La solution optimale est obtenue pour les quatre tests quand le nombre de modèles est relativement faible. Nous remarquons qu'au-delà de10 modèles, le programme linéaire ne trouve pas la solution optimale (ou n'arrive pas à prouver son optimalité) en moins de 30 minutes pour certaines instances.

L'évolution du temps de calcul en fonction du taux de charge des opérateurs est présentée dans le tableau 5.18.

Tableau 5.18 : Effet du taux de charge

Taux de charge	Temps de calcul (sec)
92,5%	463
93,6%	96
94,6%	377
95,6%	202
96,6%	128

Il n'y a pas d'évolution claire du temps de calcul en fonction du taux de charge. Nous ne pouvons pas confirmer expérimentalement que le taux de charge est un facteur de complexité.

L'analyse numérique de complexité a permis de montrer que les facteurs de complexité du modèle proposé sont le nombre de produits, le nombre de modèles et le nombre d'opérateurs. Les temps de calcul croissent lorsque ces paramètres augmentent. Par contre, nous ne pouvons pas confirmer que le taux de charge est un facteur de complexité. Nous testons dans la prochaine section les performances du modèle sur les instances industrielles.

5.4.3 Tests sur les données du cas d'étude

Dans cette partie nous testons notre modèle sur les données réelles d'une des deux lignes d'assemblage du centre de montage Volvo à Bourg en Bresse. Sur cette ligne nous avons 92 opérateurs : 77 opérateurs de type 1, 12 opérateurs de type 2 et 3 opérateurs de type 3. Pour les journées étudiées, les durées des opérations affectées aux opérateurs de type 3 sont les mêmes pour tous les véhicules. Nous ne les prendrons donc pas en compte dans nos tests puisque la répartition de la charge de ces opérateurs sera la même quel que soit l'ordre de passage des véhicules sur la ligne. Cependant, rappelons que notre modèle peut prendre en considération les opérateurs de type 3 s'ils sont affectés par des variations des temps opératoires dans de futurs équilibrages de charge. Nous testons 9 journées de production d'environ 60 produits chacune. Nous arrêtons l'optimisation au bout de trois heures de calcul. Dans tous les cas, le programme linéaire ne trouve pas la solution optimale (ou n'arrive pas à prouver son optimalité) au bout de ce temps. Dans cette section, nous analysons les performances des solutions trouvées.

5.4.3.1 Analyse des solutions en termes de sommes des retards

Nous comparons les résultats avec les retards engendrés par la procédure utilisée actuellement qui est celle du respect des contraintes d'espacement. Les gains obtenus, calculés pour le critère proposé par notre étude qui est la somme des retards cumulés, sont affichés dans le tableau 5.19. Les gains sont calculés de la manière suivante :

$$Gap = \frac{Solution\ du\ programme\ linéaire - Solution\ actuelle}{Solution\ actuelle}$$

Notre solution est 34,5% meilleure en moyenne que les solutions trouvées par la procédure actuelle. Nous obtenons des gains plus élevés pour les opérateurs de type 1 (de l'ordre de 38%) que pour les opérateurs de type 2 pour lesquels les gains sont en moyenne de 10%. Nous détériorons dans 3 journées les retards des opérateurs de type 2

pour avoir des gains plus importants pour les opérateurs de type 1 et donc un gain global plus important.

Tableau 5.19 : Gains obtenus pour les instances du cas d'étude

Instances	*Gain global*	*Gain pour opérateurs type 1*	*Gain pour opérateurs type 2*
1	39,9%	41,0%	33,3%
2	41,9%	46,5%	17,0%
3	29,2%	29,4%	28,4%
4	36,2%	40,9%	-2,6%
5	36,5%	40,8%	-10,3%
6	31,8%	34,1%	20,5%
7	29,7%	33,9%	1,3%
8	30,5%	31,5%	19,5%
9	35,2%	42,4%	-15,7%
Moyenne	**34,54%**	**37,85%**	**10,16%**

5.4.3.2 Analyse des solutions en termes de respect de contraintes

Nous avons comparé les solutions trouvées par le programme linéaire par rapport aux solutions de la procédure actuelle en termes de retards cumulés. Nous analysons dans cette partie les solutions trouvées en termes de respects des contraintes (le critère d'optimisation utilisé actuellement dans le cas d'étude, cf. chapitre 3) et en termes de retards cumulés (le critère proposé par notre étude). Le tableau 5.20 indique pour chaque jour les contraintes violées par les deux procédures (voir les explications de l'approche du *car sequencing* dans la section 2.6.1 du chapitre 2). Plusieurs non-respects peuvent se produire dans une seule position. Dans la colonne « Contraintes violées par la procédure proposée », le nombre de positions de non-respect est donné entre parenthèses.

Pour chaque position de la procédure proposée présentant des non respects de contraintes, nous comparons la somme des retards du véhicule engendrant des viols de contraintes avec celle du même véhicule dans la séquence actuelle (la position de ce véhicule dans la séquence actuelle peut être différente de sa position dans la séquence proposée). La colonne « retards améliorés » décompte le nombre de fois où on diminue la somme des retards cumulés de ce véhicule grâce à la procédure proposée. La colonne « retards détériorés» décompte le nombre fois où le retard du véhicule occasionnant des non-respects de contraintes est plus important dans la solution proposée que dans la solution actuelle.

Tableau 5.20 : Analyse des résultats des tests du cas d'étude 1/2

	Critère d'optimisation actuel		***Critère d'optimisation proposé***		
Instances	***Contraintes violées procédure actuelle***	***Contraintes violées procédure proposée***	***Retards améliorés***	***Retards détériorés***	***Gain***
1	0	4 (4)	4	0	39,9%
2	1	5 (3)	2	1	41,9%
3	0	3 (3)	2	1	29,2%
4	5	11 (5)	3	2	36,2%
5	1	12 (10)	8	2	36,5%
6	2	6 (5)	3	2	31,8%
7	2	6 (4)	3	1	29,7%
8	3	9 (7)	3	4	30,5%
9	1	9 (7)	5	2	35,2%

Nous remarquons que les solutions du programme linéaire respectent moins bien les contraintes d'espacement que les solutions de la procédure actuelle. Ceci n'est pas surprenant puisque la procédure actuelle cherche à minimiser le nombre de non respects de contraintes alors que la procédure proposée ne prend pas en compte directement cet objectif. Mais, dans la plupart des cas, la somme des retards du véhicule engendrant des viols de contraintes est moins importante dans la solution proposée que dans la solution industrielle actuelle. Par exemple, pour la deuxième instance, nous avons cinq non-respects de contraintes dans trois positions. Pour les trois véhicules dans ces positions, la procédure proposée améliore les retards de deux véhicules sur les trois. Donc, même si on ne respecte pas les contraintes, les retards sont améliorés. Ceci est dû au fait que les contraintes sont définies d'une façon empirique et qu'elles sont parfois surdimensionnées. Par exemple, une contrainte d'espacement peut être définie à 1/3 alors qu'une étude des temps opératoires montre qu'elle pourrait être à 1/2. Si on sépare deux véhicules ayant l'option qui régit cette contrainte d'une position, un non-respect sera comptabilisé alors que la situation n'engendre pourtant pas de retard important. Cependant, dans d'autres cas, les solutions proposées détériorent la somme des retards c'est-à-dire qu'on augmente la somme des retards d'un véhicule occasionnant des non-respects de contraintes. Comme les contraintes sont définies d'une façon empirique, quelques opérateurs peuvent ne pas être considérés par des contraintes. En diminuant le retard de ces opérateurs, on diminue le retard global tout en détériorant relativement d'autres opérateurs (dont ceux affectés par des contraintes dans la procédure actuelle). Nous approfondissons cette analyse dans le tableau 5.21.

Tableau 5.21 : Analyse des résultats des tests du cas d'étude 2/2

	Critère d'optimisation actuel		***Critère d'optimisation proposé***		
Instances	***Contraintes violées procédure actuelle***	***Contraintes violées procédure proposée***	***Retards de l'opérateur le plus en retard améliorés***	***Retards de l'opérateur le plus en retard détériorés***	***Gain***
1	0	4 (4)	4	0	39,9%
2	1	5 (3)	3	0	41,9%
3	0	3 (3)	3	0	29,2%
4	5	11 (5)	4	1	36,2%
5	1	12 (10)	10	0	36,5%
6	2	6 (5)	4	1	31,8%
7	2	6 (4)	3	1	29,7%
8	3	9 (7)	4	3	30,5%
9	1	9 (7)	6	1	35,2%

Dans le tableau 5.21, pour chaque position de non-respect de contraintes, nous comparons le maximum des retards sur tous les opérateurs du véhicule dans la position de non-respect avec celui du même véhicule dans la séquence actuelle. La colonne « Retards de l'opérateur le plus en retard améliorés » décompte le nombre de fois où on diminue le maximum des retards sur tous les opérateurs grâce à la solution proposée. Par exemple, pour la deuxième instance, pour chacune des trois positions de non-respect de la procédure proposée, le maximum des retards sur tous les opérateurs a été diminué grâce à la solution trouvée. La colonne « Retards de l'opérateur le plus en retard détériorés» décompte le nombre fois où le maximum des retards du véhicule occasionnant des non-respects de contraintes est plus important dans la solution proposée que dans la solution actuelle. La procédure proposée fait souvent diminuer le maximum des retards sur tous les opérateurs pour les véhicules engendrant des non respects de contraintes. Globalement, sur les neuf instances, pour les 48 véhicules violant des contraintes on améliore pour le véhicule concerné le retard le plus élevé (et donc le plus sérieux du point de vue de la qualité de la séquence) 41 fois, et on ne le détériore que 7 fois.

L'analyse des solutions trouvées par le programme linéaire en termes de respect de contraintes montre donc la limite de l'approche de « *car sequencing* ». On voit aussi que les règles de séquencement ne sont pas très bien définies puisqu'elles n'arrivent pas à cerner les vraies surcharges des opérateurs.

5.4.3.3 Evolution de la solution avec le temps de calcul

Nous analysons dans cette partie l'évolution de la solution avec le temps de calcul. Étant donné qu'un temps de calcul de trois heures est trop long pour satisfaire les exigences industrielles, nous analysons les gains obtenus à chaque heure du temps de calcul. Les résultats sont donnés dans le tableau 5.22.

Tableau 5.22 : Evolution de la solution avec le temps de calcul

Instances	***Gain 1h***	***Gain 2h***	***Gain 3h***
1	24,1%	24,1%	39,9%
2	34,9%	41,8%	41,9%
3	21,8%	27,9%	29,2%
4	28,7%	35,8%	36,2%
5	28,6%	33,6%	36,5%
6	13,6%	30,5%	31,8%
7	25,8%	29,6%	29,7%
8	23,8%	29,5%	30,5%
9	24,5%	24,8%	35,2%
Moyenne	25,09%	30,84%	34,56%

Nous remarquons qu'après une heure de calcul, le gain moyen obtenu est de 25% par rapport à la solution proposée par la procédure actuelle, mais que nous sommes encore assez loin de la solution obtenue au bout de trois heures de calcul (34% d'amélioration).

5.4.3.4 Adaptation au cas d'étude : problème de fin de journée

Les solutions trouvées par la procédure proposée ont une limite pour notre problématique industrielle : les véhicules des dernières positions de la séquence proposée sont des véhicules qui engendrent des retards importants (appelés véhicules lourds). Ceci est dû au fait que les retards des dernières positions ne sont pas répercutés sur d'autres véhicules dans notre calcul puisqu'ils sont les derniers dans la séquence calculée. Plus précisément, on observe dans les résultats des tests sur les instances réelles qu'en général les deux dernières positions sont attribuées aux véhicules les plus lourds de la journée. Mais ces retards importants seront en pratique répercutés sur les véhicules séquencés aux premières positions de la journée suivante puisque le véhicule séquencé à la dernière position sera au premier poste de travail de la ligne à la fin de la journée et sa fabrication continuera donc le lendemain sur les autres postes de travail. Ce n'est donc pas satisfaisant et nous avons modifié légèrement notre procédure pour éviter cet écueil.

La solution proposée pour remédier à ce problème est de pondérer les retards des deux dernières positions : les retards de la dernière position sont multipliés par 3 et les retards de l'avant dernière position sont multipliés par 2. Pour introduire ces poids, nous adaptons simplement la fonction objectif de notre programme linéaire qui devient $\min f = \sum_{p=1}^{P} \sum_{j=1}^{N} \pi_j w_{jp}$ avec π_j le poids de chaque position ($\pi_j = 1\ si\ j \in [1..N-2], \pi_{N-1} = 2\ et\ \pi_N = 3$). Nous analysons ensuite les solutions obtenues : les gains sont donnés dans le tableau 5.23.

Tableau 5.23 : Gains obtenus avec et sans pondération des dernières positions

Instances	***Sans pondération***	***Avec pondération***
1	39,9%	35,2%
2	41,9%	38,8%
3	29,2%	22,7%
4	36,2%	28,0%
5	36,5%	30,6%
6	31,8%	22,1%
7	29,7%	21,8%
8	30,5%	23,1%
9	35,2%	33,6%
Moyenne	34,56%	28,42%

La pondération des retards des deux dernières positions permet d'obtenir l'effet escompté : les derniers véhicules de la séquence ne sont plus des véhicules lourds. La moyenne des gains diminue mais reste élevée (plus de 28%). C'est donc une bonne solution pour remédier au problème de fin de journée.

5.5 Conclusion

Le programme linéaire proposé permet de modéliser le problème rencontré dans le cas industriel. Il prend en compte les trois types d'opérateurs de la ligne d'assemblage. Les solutions trouvées améliorent nettement le critère d'optimisation par rapport à la procédure actuelle. L'analyse de ces solutions a révélé une limite principale qui est le temps de calcul qui reste assez long. Nous testons dans les prochains chapitres d'autres méthodes de résolution.

Chapitre 6 : Résolution du problème de séquencement par la programmation dynamique

6.1 Introduction

La programmation linéaire, testée dans le chapitre précédent, présente une limite qui est le temps de calcul. Nous explorons dans ce chapitre une autre approche exacte pour résoudre le problème de séquencement d'une ligne de montage multi-modèles : la programmation dynamique. En effet, notre problème d'optimisation peut se décomposer en niveaux : la solution optimale du problème global s'exprime en fonction des solutions optimales des problèmes de taille inférieure. On peut donc trouver une formulation récursive pour notre problème. Nous proposons donc tout d'abord un programme dynamique permettant de résoudre ce problème d'une façon optimale ; il est testé sur le cas mono-opérateur et multi-opérateurs. Nous procédons aussi à une étude expérimentale de la complexité. Nous développons ensuite une heuristique basée sur le programme dynamique qu'on teste sur des données académiques puis industrielles. Une partie de ce chapitre a été publiée dans (Aroui et al., 2014b).

6.2 Formulation

Dans cette section, nous présentons la modélisation du problème de séquencement sur une ligne d'assemblage multi-modèle en utilisant la programmation dynamique. A la différence de notre programme linéaire présenté dans le chapitre 5, cette modélisation considère comme entité élémentaire le modèle et non plus le produit. Un modèle représente l'ensemble des produits ayant les mêmes temps opératoires pour chaque opérateur. Le tableau 6.1 présente un exemple de 5 produits et de 4 opérateurs. Chacun des quatre opérateurs a les même temps opérateurs pour les produits *p1*, *p3* et *p4* d'une part, et pour les produits *p2* et *p5* d'une autre part. On a donc deux modèles dans l'exemple.

Tableau 6.1 : Différence entre produit et modèle

Modèles	Produits	Opérateur 1	Opérateur 2	Opérateur 3	Opérateur 4
m1	*p1*	-0,3	0,1	-1,2	0,5
m2	*p2*	0,1	-0,6	1,3	-0,7
m1	*p3*	-0,3	0,1	-1,2	0,5
m1	*p4*	-0,3	0,1	-1,2	0,5
m2	*p5*	0,1	-0,6	1,3	-0,7

Nous utilisons les notations données dans le chapitre 4 en remplaçant l'indice du produit i par l'indice du modèle m. Par exemple, t_{mp} désigne le temps opératoire du modèle m pour l'opérateur p.

6.2.1 Formulation pour les opérateurs de type 1

Nous définissons tout d'abord le programme dynamique en prenant en compte uniquement des opérateurs de type 1 afin de simplifier la formulation. On aura donc $P = P_1$. Un programme dynamique a deux notions principales : la première est le *niveau* qui correspond à un point de décision. Dans notre cas, le niveau *j* représente la *j*^{ème} position de la séquence. La décision à prendre à ce niveau est quel modèle attribuer à la *j*^{ème} position. La deuxième notion est l'*état*. À chaque niveau *j* est associé un ensemble d'états, S_j, qui donnent les informations nécessaires pour prendre la décision, c'est-à-dire ici :

- le nombre de produits restants à séquencer de chaque modèle
- le retard cumulé de chaque opérateur après le passage de *j* produits

Dans cette section, nous prenons en compte uniquement des opérateurs de type 1. On définit donc un état *k* du niveau *j*, noté $\vec{s}_j^k$, comme un vecteur de taille $M + P$:

$$\vec{s}_j^k = \begin{pmatrix} n_1^{j,k} \\ \vdots \\ n_M^{j,k} \\ w_1^{j,k} \\ \vdots \\ w_P^{j,k} \end{pmatrix} \qquad (6.1)$$

Avec :

$n_m^{j,k}$ le nombre de produits restants à séquencer du modèle m $(\forall m = 1, \dots, M)$ pour l'état *k* du niveau *j*, c'est-à-dire la *j*^{ème} position de la séquence.

$w_p^{j,k}$ le retard cumulé de l'opérateur $p, \forall p = 1, \dots, P$ pour l'état *k* du niveau *j*. On peut exprimer le retard cumulé au niveau *j* en fonction du retard cumulé au niveau précédent : $w_p^{j,k} = \max\left(0, w_p^{j-1,k'} + \sum_{m=1}^{M} \mathbf{1}_{n_m^j} . d_{mp}\right)$ avec $w_p^{j-1,k'}$ retard de l'état $\vec{s}_{j-1}^{k'}$, prédécesseur de l'état $\vec{s}_j^k$ et $\mathbf{1}_{n_m^j}$ est une fonction indicatrice qui prend la valeur 1 si $n_m^{j,k} = n_m^{j-1,k'} - 1$ (c'est-à-dire qu'un produit du modèle *m* est séquencé à la position *j*) et 0 sinon.

Notons :

$Q(\vec{s}_j^k)$ l'ensemble des états du niveau $j-1$, prédécesseurs de l'état $\vec{s}_j^k$,

$D(\vec{s}_{j-1}^{k'}, \vec{s}_j^k)$ le retard total cumulé de tous les opérateurs à l'état k du niveau j si le prédécesseur est l'état k' du niveau $j-1$, $\vec{s}_{j-1}^{k'} \in Q(\vec{s}_j^k)$.

L'équation (6.2) illustre la façon de calculer $D(\vec{s}_{j-1}^{k'}, \vec{s}_j^k)$:

$$D(\vec{s}_{j-1}^{k'}, \vec{s}_j^k) = \sum_{p=1}^{P} w_p^{j,k} \qquad (6.2)$$

En réutilisant la définition de $w_p^{j,k}$ donné plus haut, on obtient :

$$D(\vec{s}_{j-1}^{k'}, \vec{s}_j^k) = \sum_{p=1}^{P} \max\left(0, w_p^{j-1,k'} + \sum_{m=1}^{M} \mathbf{1}_{n_m^j} . d_{mp}\right)$$

Notons :

W_j^k la somme des retards cumulés de tous les opérateurs sur tous les produits séquencés depuis le niveau 0 au niveau j (c'est-à-dire la séquence des j premiers éléments) à l'état k du niveau j.

La différence entre $D(\vec{s}_{j-1}^{k'}, \vec{s}_j^k)$ et W_j^k est que le premier est le retard total cumulé de tous les opérateurs à l'état k du niveau j alors que le deuxième est la somme de ces retards du niveau 0 au niveau j.

La somme des retards cumulés W_j^k est écrite d'une manière récursive comme suit :

$$W_j^k = \min_{\forall\, \vec{s}_{j-1}^{k'} \in Q(\vec{s}_j^k)} \left\{W_{j-1}^{k'} + D(\vec{s}_{j-1}^{k'}, \vec{s}_j^k)\right\}, \qquad \forall\, j \neq 0, \forall\, k$$
$$W_0^1 = 0\,, \text{ pour } \vec{s}_0^1 \qquad (6.3)$$

La fonction récursive (6.3) définit ainsi un programme dynamique.

Le graphe de la figure 6.1 illustre notre modèle de programmation dynamique sur un exemple comprenant un seul opérateur de type 1, 5 produits répartis en deux modèles (2 produits du modèle 1 et 3 produits du modèle 2). On a donc $P = P_1 = 1, N = 5, M = 2$, $N_1 = 2$ et $N_2 = 3$. Le modèle 1 entraine un temps de repos de 3 unités de temps pour l'opérateur ($d_{11} = -3$) et le modèle 2 engendre un retard d'une unité de temps pour l'opérateur ($d_{21} = 1$).

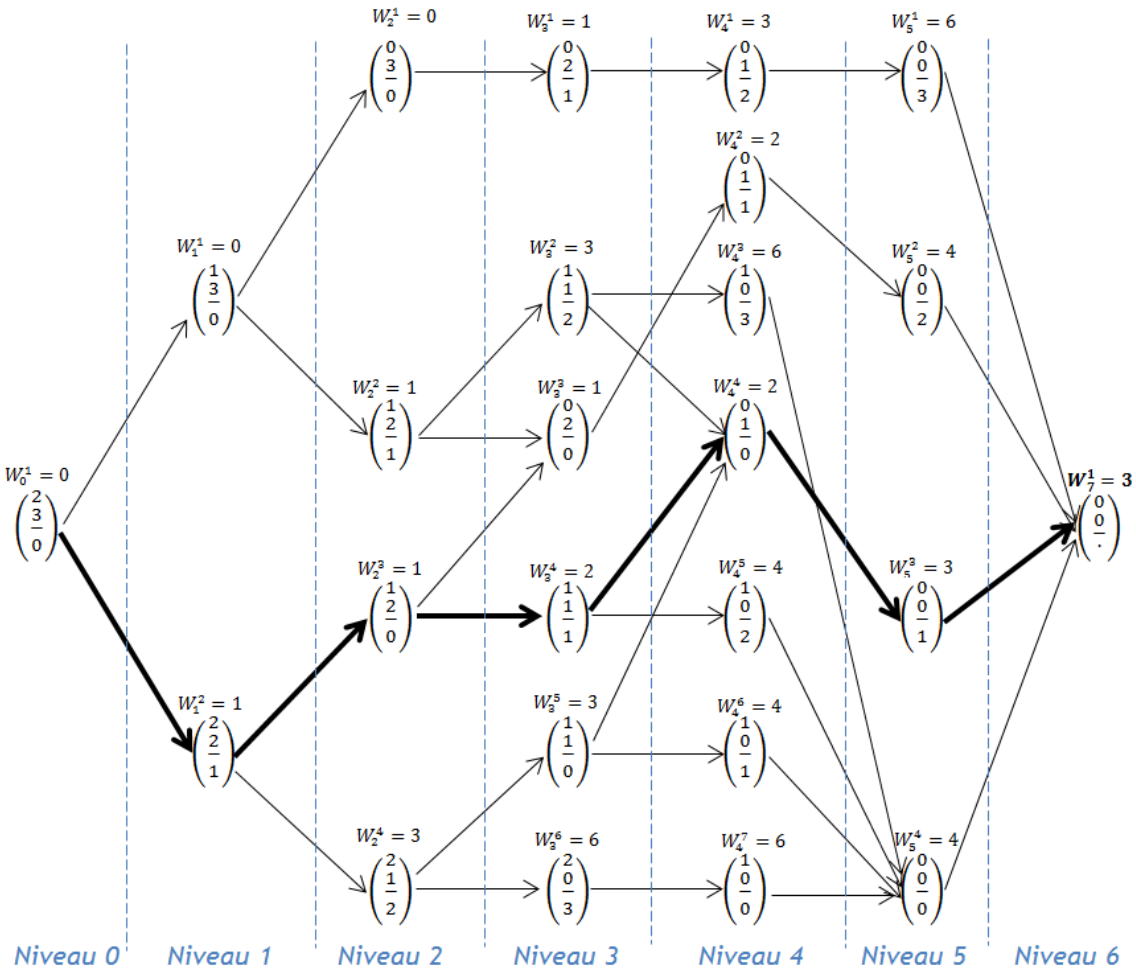


Figure 6.1 Graphe d'un exemple d'un opérateur de type 1

Chaque nœud du graphe symbolise un état $\vec{s}_j^k$. L'état $\vec{s}_0^1 = (2\,,3|0)$ est le seul nœud du niveau 0. Il représente l'état initial du programme dynamique : les produits à séquencer (2 produits du modèle 1 et 3 produits du modèle 2) et l'état initial du retard. Ici le retard initial $w_1^{0,1}$ est égal à 0. A chaque état, on a au maximum M choix (2 dans notre exemple). Par exemple, au nœud (2 ,3|0), on peut choisir de séquencer un produit du modèle 1 à la première position, ce qui génère le nœud (1 ,3|.) ; ou on peut choisir d'attribuer un produit du modèle 2 à la première position, ce qui génère le nœud (2 ,2|.). Pour chaque état généré, nous calculons le retard cumulé de chaque opérateur (un seul dans notre exemple). Par exemple, pour le nœud généré (2, 2|.), le retard cumulé est $w_1^{1,2} = \max\left(0, w_1^{0,1} + 1\right) = 1$. L'état devient alors (2, 2| 1).

A chaque état, on utilise la formule récursive (6.3) pour calculer W_j^k. Par exemple, l'état $\vec{s}_4^4$ au niveau 4 a trois prédécesseurs $\vec{s}_3^2$, $\vec{s}_3^4$ et $\vec{s}_3^5$. W_4^4 est alors déterminé comme suit :

$$W_4^4 = \min\{W_3^2 + D(\vec{s}_3^2, \vec{s}_4^4)\ ;\ W_3^4 + D(\vec{s}_3^4, \vec{s}_4^4)\ ;\ W_3^5 + D(\vec{s}_3^5, \vec{s}_4^4)\}$$

$$= \min\{3 + \max(0; 2-3)\ ;\quad 2 + \max(0; 1-3)\ \ ;\quad 3 + \max(0; 0-3)\} = 2$$

Le minimum est obtenu en séquençant un produit du premier modèle à partir du prédécesseur $\vec{s}_3^4$. Nous marquons $\vec{s}_3^4$ comme le meilleur prédécesseur de $\vec{s}_4^4$. On répète cette procédure avec chaque état généré, jusqu'à ce qu'aucun produit ne reste à séquencer c'est-à-dire jusqu'au dernier nœud du graphe (0, 0| .). Le retard du dernier nœud est le retard total de la meilleure séquence. Dans notre exemple, le retard du dernier nœud du graphe est $W_1^7 = 3$. Il correspond au retard total de la meilleure séquence qu'on trouve en utilisant les informations du meilleur prédécesseur de chaque nœud. Le meilleur prédécesseur de $\vec{s}_6^1$ est $\vec{s}_5^3$, son meilleur prédécesseur est $\vec{s}_4^4$… La meilleure séquence de notre exemple, dessinée en gras, est donc : 2-1-2-1-2.

6.2.2 Formulation pour les opérateurs de type 2 et 3

Afin d'étendre la formulation précédente aux opérateurs type 2 et type 3, nous ajoutons deux paramètres au vecteur caractérisant un état :

$$\vec{s}_j^k = \begin{pmatrix} n_1^{j,k} \\ \vdots \\ n_M^{j,k} \\ \hline r_1^{j,k} \\ \vdots \\ r_{P_2}^{j,k} \\ \hline u_1^{j,k} \\ \vdots \\ u_{P_3}^{j,k} \\ \hline w_1^{j,k} \\ \vdots \\ w_P^{j,k} \end{pmatrix} \qquad (6.4)$$

Avec :

$r_p^{j,k}$ une variable intermédiaire pour le calcul du retard de l'opérateur p de type 2, $\forall p = 1, \dots, P_2$ pour l'état k du niveau j. C'est le retard de l'opérateur p s'il n'avait qu'un temps de cycle pour finir ses opérations.

$u_p^{j,k}$ une variable intermédiaire pour le calcul du retard de l'opérateur p de type 3, $\forall p = 1, \dots, P_3$ pour l'état k du niveau j. Elle mémorise le retard qu'a eu l'opérateur p de type 3 sur le dernier véhicule qu'il a traité afin de le comptabiliser sur le prochain à traiter.

Nous rappelons les notations des données utilisés pour les opérateurs de type 2 et 3 :

$\propto_{mp}^{2}$ variable binaire qui est égale à 1 si l'opérateur p de type 2 doit effectuer des opérations sur un produit du modèle m et 0 sinon, $\forall m = 1, \dots, M, \forall p = 1, \dots, P_2$

β_{mp}^{2} nombre de temps de cycle pendant lesquels l'opérateur p de type 2 peut travailler sur un produit du modèle m, $\forall m = 1, \dots, M, \forall p = 1, \dots, P_2$

$\propto_{jp}^{3}$ paramètre binaire qui est égal à 1 si l'opérateur p de type 3 doit effectuer des opérations sur le produit de la position j et 0 sinon, $\forall j = 1, \dots, N, \forall p = 1, \dots, P_3$

β_{p}^{3} nombre de temps de cycle pendant lesquels l'opérateur p de type 3 peut travailler sur un produit, $\forall p = 1, \dots, P_3$

Nous adaptons les formules introduites pour le calcul des retards cumulés dans la programmation linéaire (chapitre 5, section 5.2.2) en utilisant les temps opératoires des modèles plutôt que ceux des produits. Pour les opérateurs de type 2, le retard se calcule de la façon suivante :

$$w_p^{j,k} = \begin{cases} \max\left(r_p^{j,k} - \left(\sum_{m=1}^{M} \mathbf{1}_{n_m^j} . \beta_{mp}^{2} - 1\right) * \gamma \ , \ \ 0\right) & \text{si} \quad \propto_{mp}^{2} = 1 \\ 0 & \text{sinon} \end{cases}$$

Avec : $r_p^{j,k} = \max\left(0, r_p^{j-1,k'} + \sum_{m=1}^{M} \mathbf{1}_{n_m^j} . d_{mp}\right), \forall\, \vec{s}_{j-1}^{\,k'} \in Q\left(\vec{s}_j^{\,k}\right)$

Les retards des opérateurs de type 3 se calculent comme suit :

$$w_p^{j,k} = \begin{cases} \max\left(0, u_p^{j-1,k} + \sum_{m=1}^{M} \mathbf{1}_{n_m^j} . d_{mp} - \left(\sum_{m=1}^{M} \mathbf{1}_{n_m^j} . \beta_{p}^{3} - 1\right) * \gamma \ , \ \ 0\right) & \text{si} \quad \propto_{jp}^{3} = 1 \\ 0 & \text{sinon} \end{cases}$$

Avec : $u_p^{j,k} = \begin{cases} w_p^{j,k} & \text{si} \quad \propto_{jp}^{3} = 1 \\ w_p^{j-1,k'} & \text{sinon} \end{cases}, \forall\, \vec{s}_{j-1}^{\,k'} \in Q\left(\vec{s}_j^{\,k}\right)$

$u_p^{j,k}$ mémorise le retard qu'a eu l'opérateur p de type 3 sur le dernier véhicule qu'il a traité. On actualise sa valeur à chaque fois que l'opérateur p traite un véhicule ($\propto_{jp}^{3} = 1$).

La somme des retards cumulés W_j^k se calcule de manière récursive comme indiqué dans l'équation (6.3) dont nous rappelons ici la formulation :

$$W_j^k = \min_{\forall\, \vec{s}_{j-1}^{\,k'} \in Q\left(\vec{s}_j^{\,k}\right)} \left\{W_{j-1}^{k'} + D\left(\vec{s}_{j-1}^{\,k'}, \vec{s}_j^{\,k}\right)\right\}, \qquad \forall\, j \neq 0, \forall\, k, W_0^1 = 0 \qquad (6.3)$$

Avec $D\left(\vec{s}_{j-1}^{k'}, \vec{s}_j^{k}\right) = \sum_{p=1}^{P} w_p^{j,k}$

La figure 6.2 présente le graphe sur lequel s'appuie le programme dynamique pour un exemple avec 4 opérateurs (1 opérateur de type 1, 1 opérateur de type 2 et 2 opérateurs de type 3) et 5 produits répartis en deux modèles (2 produits du modèle 1 et 3 produits du modèle 2). Le tableau 6.2 indique les temps opératoires de chaque opérateur.

L'opérateur 2 nécessite deux temps de cycle pour exécuter ses opérations sur le modèle 1. Les deux opérateurs de type 3 alternent les véhicules. L'opérateur 3 travaillera sur les véhicules en position 1, 3 et 5 alors que l'opérateur 4 travaillera sur les véhicules en position 2 et 4. Le temps de cycle est 10 minutes.

Tableau 6.2 : Temps opératoires de l'exemple

Opérateur	*Opérateur type 1*	*Opérateurs type 2*	*Opérateurs type 3*	
	1	2	3	4
m1	11	21 (2)	18	18
m2	9	0	21	21

Vu les données de notre exemple, la structure d'un état à une position *j* donnée est la suivante : les deux premières valeurs représentent le nombre de produits restants à séquencer de chaque modèle à la position *j*, les trois valeurs suivantes représentent les variables intermédiaires pour le calcul des retards des opérateurs de type 2 et 3 et les quatre dernières valeurs indiquent le retard cumulé à la position *j* de chaque opérateur. Comme pour le graphe de l'exemple avec l'opérateur de type 1, chaque nœud du graphe symbolise un état $\vec{s}_j^{k}$. L'état $\vec{s}_0^{1}$ représente l'état initial du programme dynamique. Depuis l'état initial, on a le choix de mettre un produit du modèle 1 ou bien du modèle 2 dans la première position de la séquence. Le premier choix engendre un retard d'une unité de temps pour les opérateurs 1 et 2. L'opérateur 3 prend en compte le produit en première position. Si le modèle 1 est choisi, ceci n'engendre pas de retard (18 minutes d'opérations sur les 20 minutes disponibles). L'opérateur 4 n'a pas de tâches à effectuer donc aura un retard nul. Si à partir de ce nœud on choisit d'attribuer la deuxième position à un produit du modèle 2, les opérateurs 2 et 3 n'ont pas de tâches à effectuer et donc n'ont pas de retard sur ces véhicules. Les variables intermédiaires $r_p^{j,k}$ et $u_p^{j,k}$ permettent de garder en mémoire le retard qu'avaient ces opérateurs sur le véhicule précédent.

Dans l'exemple, le retard du dernier nœud du graphe est $W_6^1 = 8$. Il correspond au retard total de la meilleure séquence qu'on trouve en utilisant les informations de meilleur prédécesseur de chaque nœud. L'exemple a plusieurs solutions optimales. Une des séquences optimale, montrée en gras dans la figure 6.2, est 1-2-2-2-1.

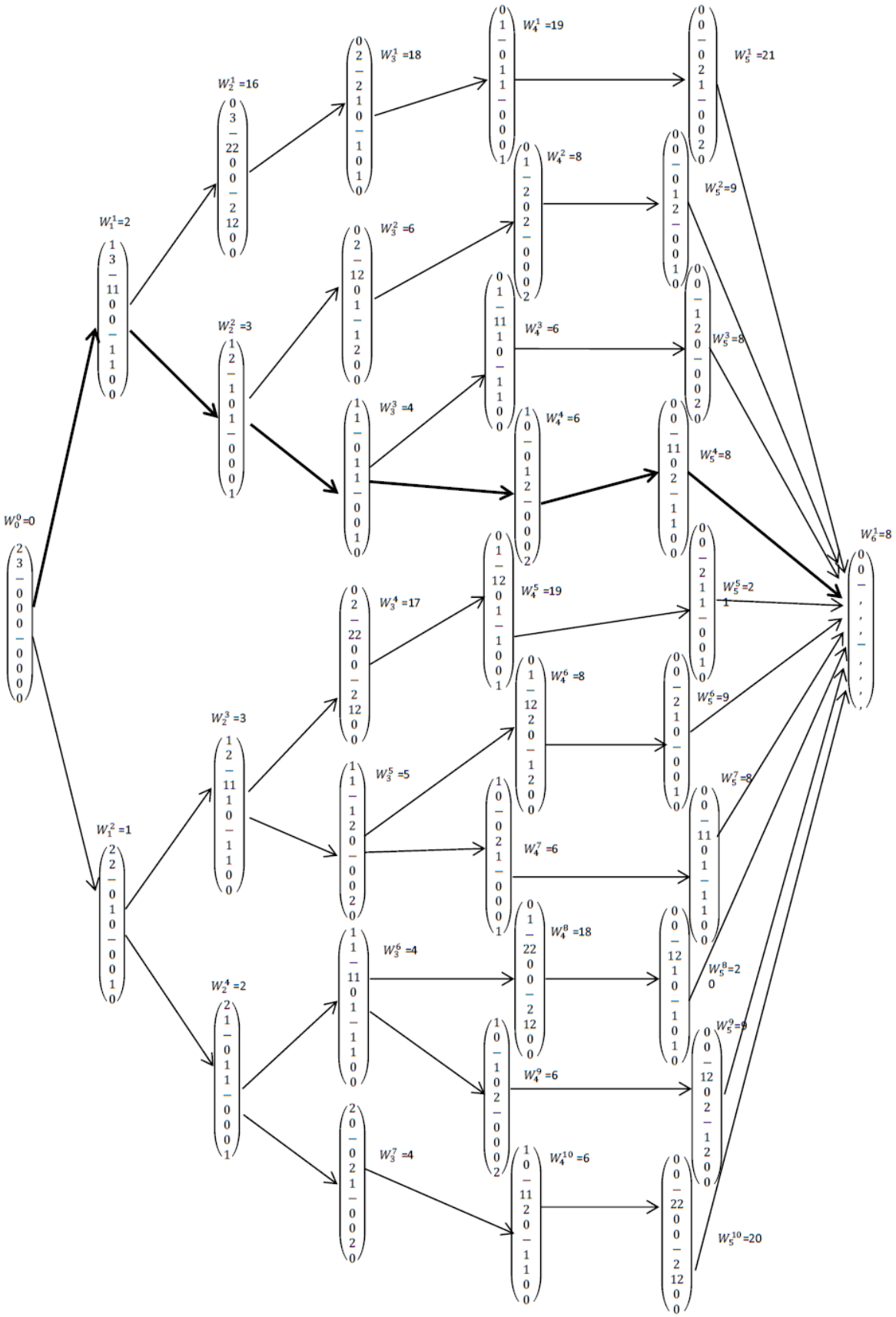


Figure 6.2 Graphe d'un exemple avec tous les types d'opérateurs

6.3 Algorithme

Le modèle de programmation dynamique présenté dans la section 6.2 a été programmé en langage Java. À partir de la formulation décrite dans la section précédente quelques choix ont été faits pour optimiser l'utilisation de la mémoire et pour gagner en temps de calcul. Un de ces choix est que l'algorithme privilégie l'exploration des nœuds en profondeur c'est-à-dire qu'il privilégie d'avancer dans les niveaux (de gauche à droite sur les figures 6.1 et 6.2) plutôt qu'explorer des états du même niveau (de haut en bas sur les figures 6.1 et 6.2).

Lors du parcours du graphe, on garde en mémoire la meilleure solution déjà trouvée qu'on appelle $Referencedelay$. Nous utilisons les paramètres suivants pour chaque état :

- *Best Partial Delay,* BPD_j^k : la somme des retards cumulés de l'actuel meilleur chemin pour atteindre la fin du graphe à partir de l'état k du niveau j $\vec{s}_j^k$.

- *Partial Lower Bound,* PLB_j^k : une borne inférieure de BPD_j^k. Elle est calculée par la somme des retards (et non pas des repos) des véhicules restants à séquencer. Ceci correspond à la somme des parties positives des d_{mp}. Nous rappelons que $d_{mp} = t_{mp} - \gamma$ est un retard s'il est positif et un repos s'il est négatif. L'algorithme de notre programme dynamique comprend un test basé sur cette borne inférieure qui permet de juger le potentiel d'un état à améliorer la solution actuelle du programme.

Algorithme de recherche en profondeur :

A. Initialiser le retard de référence
B. **Procédure Optimal_Schedule ($\vec{s}_0^0$)**

À l'étape A, nous cherchons une solution initiale au problème en menant une recherche en profondeur jusqu'à arriver au dernier nœud du graphe. Nous assignons la valeur de cette solution au retard de référence.

À l'étape B, nous appliquons la procédure *Optimal_Schedule* pour explorer le premier état du graphe $\vec{s}_0^0$ pour trouver le meilleur chemin à partir de l'état $\vec{s}_0^0$ jusqu'au nœud final du graphe et donc trouver la solution optimale au problème.

La procédure Optimal_Schedule ($\vec{s}_j^{k'}$), détaillée ci-après, évalue le chemin à partir de l'état $\vec{s}_j^{k'}$ jusqu'au nœud final du graphe d'une façon à privilégier l'exploration en

profondeur c'est-à-dire qu'on privilégie d'avancer dans les niveaux plutôt qu'explorer des états du même niveau.

Le pseudo code de cette procédure est donné ci-dessous.

Procédure Optimal Schedule ($\vec{s}_j^{k'}$) :

Entrées : $\vec{s}_j^{k'}$ (l'état k' du niveau j) et le retard de référence du graphe

I. Si ($\sum_{m=1}^{M} n_m^{j,k'}$>0) alors

{

Pour m=1 jusqu'à M (M nombre de modèles)

{

Si ($n_m^{j,k'} \neq 0$) alors

{

a. Créer (si pas encore créé) l'état succédant $\vec{s}_{j+1}^{k}$ si on ajoute un produit du modèle *m* à la position *j+1* en partant de l'état $\vec{s}_j^{k'}$: $n_m^{j+1,k} = n_m^{j,k'} - 1$

b. Si $W_j^{k'} + \sum_{p=1}^{P} w_p^{j+1,k} \leq W_{j+1}^{k}$ alors

{

b.1. $W_{j+1}^{k} \leftarrow W_j^{k'} + \sum_{p=1}^{P} w_p^{j+1,k}$

b.2. Calculer PLB_{j+1}^{k}

Si $W_{j+1}^{k} + PLB_{j+1}^{k} < Retard\ de\ référence$ alors

Procédure **Optimal_Schedule (**entrée: $\vec{s}_{j+1}^{k}$)

b.3. Mettre à jour le *Best partial delay* de l'état *k'*:

Si $\sum_{p=1}^{P} w_p^{j+1,k} + BPD_{j+1}^{k} < BPD_j^{k'}$ Alors

{

$BPD_j^{k'} \leftarrow \sum_{p=1}^{P} w_p^{j+1,k} + BPD_{j+1}^{k}$

Mettre m comme meilleur modèle suivant à *k'*

}

}

}

}

}

II. Mettre à jour le retard de référence du graphe :

SI $BPD_j^{k'} + W_j^{k'} < Referencedelay$ alors

$Referencedelay \leftarrow BPD_j^{k'} + W_j^{k'}$

La procédure comprend un test de borne inférieure. Ceci consiste à calculer une borne inférieure du chemin partant de l'état $\vec{s}_j^{k'}$ jusqu'au nœud final du graphe et à vérifier si le chemin peut potentiellement améliorer la solution actuelle ($Referencedelay$). Si le chemin est exploré, et qu'on arrive au dernier nœud du graphe, une nouvelle solution est trouvée. On la compare au retard de référence du graphe qui est mis à jour si la solution trouvée est meilleure. La procédure s'arrête quand tous les successeurs de l'état $\vec{s}_j^{k'}$ sont parcourus. Le *Best Partial Delay* $BPD_j^{k'}$ donne alors le retard cumulé du meilleur chemin pour atteindre la fin du graphe à partir de l'état $\vec{s}_j^{k'}$.

La procédure parcourt tous les modèles pour voir s'il reste des produits à séquencer. À l'étape a, un des successeurs de l'état $\vec{s}_j^{k'}$ est créé (sauf si déjà créé). Notons $\vec{s}_{j+1}^{k}$ un des successeurs de $\vec{s}_j^{k'}$ résultant de l'attribution d'un produit du modèle m à la position $j+1$. Les premiers M éléments de $\vec{s}_{j+1}^{k}$ s'écrivent alors comme suit :

$$n_1^{j+1,k} = n_1^{j,k'},\ \ n_2^{j+1,k} = n_2^{j,k'}, \dots, n_m^{j+1,k} = n_m^{j,k'} - 1, \dots,\ \ n_M^{j+1,k} = n_M^{j,k'}$$

Si $\vec{s}_{j+1}^{k}$n'existait pas, son retard cumulé W_{j+1}^{k} est initialisé à l'infini. Dans le cas contraire, W_{j+1}^{k} a été déjà calculé grâce à un autre chemin. Il faut vérifier si on améliore W_{j+1}^{k} en arrivant par $\vec{s}_j^{k'}$. Ceci est fait aux étapes b et b.1 par application de l'équation (6.3) de la formulation. Si on ne change pas la valeur de W_{j+1}^{k}, il est inutile d'explorer $\vec{s}_{j+1}^{k}$, i.e. rechercher à nouveau le meilleur chemin issu de cet état jusqu'à la fin du graphe.

On calcule une borne inférieure PLB_{j+1}^{k}au retard du chemin partant de $\vec{s}_{j+1}^{k}$ et atteignant la fin du graphe. Cette borne est la somme des retards des produits restants à séquencer. C'est la somme des parties positives des d_{mp} des produits restants. À l'étape b.2, on vérifie s'il est intéressant d'explorer $\vec{s}_{j+1}^{k}$. On compare la somme de la borne inférieure et du retard cumulé au retard de référence. Si l'inéquation est vérifiée, $\vec{s}_{j+1}^{k}$ peut faire partie d'un chemin meilleur que celui du retard de référence. On applique alors la procédure *Optimal_Schedule* pour éventuellement parcourir ce chemin.

À l'étape b.3, on met à jour le *Best Partial Delay*, BPD_j^{k} de l'état $\vec{s}_j^{k'}$ en considérant celui de son successeur $\vec{s}_{j+1}^{k}$. Si BPD_j^{k} est mis à jour grâce à $\vec{s}_{j+1}^{k}$, on garde cette trace et on enregistre que l'actuel meilleur modèle à séquencer quand on est à $\vec{s}_j^{k'}$ est le modèle m.

À l'étape II, on vérifie si on a trouvé une meilleure solution pour notre graphe. Si c'est le cas, on met à jour le retard de référence. Cette procédure est pour un état donné $\vec{s}_j^{k'}$. Si on l'applique à l'état $\vec{s}_0^{0}$, elle nous donne la solution optimale au problème complet.

Ce programme dynamique sera testé sur des instances mono-opérateur et des instances multi-opérateurs dans la section suivante.

6.4 Étude numérique pour le cas mono-opérateur

Dans cette partie, nous analysons expérimentalement les effets du nombre de produits et du nombre de modèles sur le temps de calcul. Nous utilisons les instances générées pour l'expérimentation du programme linéaire (section 5.3.1, chapitre 5). Le temps maximal de calcul est 10 minutes. Nous notons les instances pour lesquelles la solution optimale a été obtenue sans dépasser cette limite. Les résultats de cette analyse sont exposés dans le tableau 6.3. Chaque cellule indique combien d'instances fournissent la solution optimale dans la limite de temps fixée parmi les 20 instances générées.

Par exemple pour 40 produits et pour un nombre de modèles qui varie entre 5 et 6, la solution optimale a été obtenue pour 18 instances sur les 20 générées en moins de 10 minutes.

Tableau 6.3 : Analyse numérique de complexité du cas mono-opérateur

Nombre de produits / **Segment de nombre de modèles**	**20 produits**	**30 produits**	**40 produits**	**50 produits**	**60 produits**
18-29 modèles		17/20	14/20	14/20	13/20
12-16 modèles	19/20	18/20	16/20	13/20	18/20
8-11 modèles	20/20	16/20	12/20	10/20	13/20
5-6 modèles	20/20	13/20	18/20	12/20	12/20

Nous remarquons qu'il n'y a pas d'évolution claire du nombre des instances résolues en moins de 10 minutes en fonction du nombre de modèles et du nombre de produits. Nous ne pouvons donc pas établir la complexité numérique de cette approche en fonction du nombre de modèles et du nombre de produits. Nous observons aussi que les résultats obtenus sont nettement meilleurs que ceux obtenus avec la programmation linéaire pour les mêmes instances. Le programme dynamique trouve la solution optimale en moins de 10 minutes pour 15 instances sur 20 en moyenne alors que le programme linéaire trouve la solution optimale (et prouve son optimalité) pour 6 instances sur 20 en moyenne.

6.5 Étude numérique pour le cas multi-opérateur

Dans cette section, on s'intéresse au problème prenant en compte plusieurs opérateurs. Tout d'abord, nous testons l'effet de la borne inférieure sur le temps de calcul. Ensuite,

nous procédons à des tests académiques pour valider notre modèle. Enfin, nous analysons expérimentalement la complexité de notre modèle en se basant sur les temps de calcul.

6.5.1 Effet de la borne inférieure

L'algorithme de notre programme dynamique comprend un test basé sur une borne inférieure qui permet de juger le potentiel d'un état à améliorer la solution actuelle du programme. PLB_j^k est une borne inférieure du retard du chemin partant de l'état $\vec{s}_j^k$ jusqu'au nœud final du graphe. Elle est la somme des retards (et non pas des repos) des véhicules restants à séquencer. Ceci correspond à la somme des parties positives des d_{mp}. Nous rappelons que $d_{mp} = t_{mp} - \gamma$ est un retard si d_{mp} est positif et un repos si d_{mp} est négatif.

Nous rappelons que W_j^k est la valeur de l'objectif (la somme des retards cumulés) du début du graphe jusqu'à l'état $\vec{s}_j^k$. Si $W_j^k + PLB_j^k < ReferenceDelay$ alors l'état sera exploré. Ceci veut dire que si la somme du retard du début du graphe jusqu'à l'état $\vec{s}_j^k$ d'une part, et de la borne inférieure du retard de la branche partant de cet état jusqu'au nœud final du graphe d'autre part, est inférieure à la solution actuelle, l'état $\vec{s}_j^k$ est exploré.

Cette borne est intéressante dans le cas mono-opérateur puisque la somme des parties positives des d_{mp} se rapproche de la solution optimale si on a des véhicules qui permettent de rattraper systématiquement le retard. Nous essayons d'expérimenter si cette borne inférieure est intéressante pour le cas multi-opérateurs c'est-à-dire si elle permet de gagner en temps de calcul. Nous utilisons trois instances multi-opérateurs (de type 1) que nous résolvons avec le programme dynamique sans le test basé sur la borne inférieure en premier lieu, et avec ce test en second lieu (deuxième ligne de l'étape b.2 dans l'algorithme). Nous limitons le temps de calcul à 30 minutes. Les temps de calcul sont donnés dans le tableau 6.4.

Tableau 6.4 : Effet de la borne inférieure

Instance	Nombre de produits	Nombre d'opérateurs	Temps de calcul (s)	
			Procédure sans la borne inférieure	Procédure avec la borne inférieure
1	22	4	853	226
2	22	4	369	87
3	24	4	*1800	425

La procédure sans la borne inférieure ne trouve pas la solution optimale au bout de 30 minutes de calcul pour la troisième instance indiquée par (*). Nous observons plus généralement pour les trois instances que la borne inférieure est intéressante puisqu'elle permet de gagner en temps de calcul. Elle sera donc utilisée dans la suite des tests. Dans la prochaine section, nous analysons numériquement la complexité du programme numérique afin de déterminer ses limites et ses facteurs de complexité.

6.5.2 Tests académiques

Nous testons le programme dynamique sur les opérateurs de type 2 et de type 3. Ceci nous permet de valider notre modèle au sens qu'il permet bien de trouver la solution optimale. Pour ceci, nous utilisons les instances académiques utilisées pour tester le programme linéaire dans le chapitre 5 (section 5.4.1.2). La solution optimale, évidement identique à celle fournie par le programme linéaire, est obtenue en moins d'une seconde pour chacune des deux premières instances et au bout de 2 minutes et 31 secondes de calcul pour la troisième instance. Les temps de calcul sont similaires à ceux du programme linéaire pour les deux premières instances. Pour la troisième instance, le temps de calcul est plus long que celui du programme linéaire.

6.5.3 Analyse numérique de la complexité

Dans cette partie, nous analysons expérimentalement les limites du programme dynamique en utilisant des instances multi-opérateurs. Nous considérons un cas de base conçu à partir des données de notre cas d'étude. Ces données ont les paramètres suivants : 4 opérateurs de type 1, 20 produits, 5 modèles et un taux de charge moyen de 92,5%. Ce cas de base est celui utilisé pour les instances multi-opérateurs pour l'analyse numérique de la complexité du programme linéaire (tableau 5.14). À partir de ce cas de base, nous avons généré des instances afin d'analyser numériquement la complexité du programme linéaire dans le chapitre précédent (section 5.4.2). Ces instances permettent d'augmenter progressivement un des paramètres tout en gardant les autres constants. Nous limitons le temps de calcul à 30 minutes. Le but est d'analyser numériquement la complexité du programme dynamique proposé mais aussi de comparer ses performances à celles du programme linéaire.

Dans le tableau 6.5, nous expérimentons l'effet du nombre d'opérateurs sur le temps de calcul. Tous les opérateurs sont du type 1. Chaque ligne du tableau représente la moyenne du temps de calcul de tests sur quatre différentes instances avec les mêmes paramètres. Nous testons les instances utilisées pour analyser la complexité du programme linéaire. Cependant, la limite du programme dynamique n'est pas atteinte

avec ces instances. Nous générons alors des instances supplémentaires afin d'augmenter davantage le nombre d'opérateurs.

Le signe (*) indique qu'au moins un des quatre tests n'atteignent pas la solution optimale au bout de 30 minutes de calcul. Dans ce cas, la valeur donnée est une borne inférieure au temps de calcul puisque nous utilisons 1800 secondes (30 minutes) pour ces cas-là dans le calcul de la valeur moyenne.

Dans le tableau 6.5, on observe que la limite du programme dynamique est atteinte à partir de 32 opérateurs alors que celle du programme linéaire est atteinte à partir de 16 opérateurs (voir tableau 5.15). Par exemple, le programme linéaire ne donne pas la solution optimale pour les quatre instances à 24 opérateurs au bout de 30 minutes de calcul alors que le programme dynamique trouve les solutions optimales des quatre instances en 41 secondes en moyenne. Le programme dynamique est donc plus performant que le programme linéaire pour ces instances.

Tableau 6.5 : Effet du nombre d'opérateurs à séquencer

Nombre des opérateurs	Temps de calcul (sec)
4	42
8	8
12	8
16	15
20	23
24	41
32	*534
40	*645
48	*1800

Le tableau 6.6 présente l'évolution du temps de calcul en fonction du nombre de produits. Nous remarquons que le temps de calcul croît considérablement avec le nombre de produits.

Tableau 6.6 : Effet du nombre de produits à séquencer

Nombre de produits	Temps de calcul (sec)
20	42
22	200
24	*1394
26	*1800
28	*1800
30	*1800

Dès 24 produits, les limites du programme dynamique commencent à apparaitre (contre 22 produits pour le programme linéaire). Nous notons aussi qu'à partir de 26 produits le programme dynamique ne donne aucune solution optimale en moins de 30 minutes de calcul pour les instances testées. Les performances des programmes linéaire et dynamique sont assez proches.

Dans le tableau 6.7, nous présentons l'évolution du temps de calcul en fonction du nombre de modèles. La solution optimale est obtenue pour les quatre tests quand le nombre de modèles est relativement faible. Nous remarquons que lorsqu'on dépasse 10 modèles, la procédure ne trouve pas la solution optimale en moins de 30 minutes pour certaines instances. La performance du programme dynamique est un peu meilleure que celle du programme linéaire qui ne trouve pas la solution optimale pour toutes les instances testées au-delà de 10 modèles.

Tableau 6.7 : Effet du nombre de modèles

Nombre de modèles	Temps de calcul (sec)
5	42
10	*629
15	*954
20	*696

L'analyse numérique de complexité permet ainsi de montrer que les facteurs de complexité du programme dynamique proposé sont le nombre de produits et le nombre des opérateurs. Le temps de calcul croît lorsqu'on augmente ces paramètres. Par contre, nous ne pouvons pas confirmer que le nombre de modèles est un facteur de complexité puisque l'évolution du temps de calcul n'est pas claire en fonction de ce facteur. De plus, cette analyse montre que les performances du programme dynamique sont un peu meilleures que celles du programme linéaire pour les mêmes instances.

En outre, cette analyse met en évidence que les limites du programme dynamique apparaissent à partir de 24 produits (si on espère avoir une solution optimale en moins de 30 minutes de calcul) ce qui est inapproprié à l'application au cas industriel où le nombre de produits peut aller de 50 à 200 produits par jour. Afin de remédier à ces limites, nous développons dans la section suivante une heuristique basée sur le programme dynamique.

6.6 Heuristique

L'étude expérimentale du programme dynamique a montré les limites de cette méthode exacte qui s'avère inappropriée à l'application au cas industriel. Cependant, nous pouvons profiter de la structure du graphe pour concevoir des méthodes

heuristiques. Dans cette section, nous présentons une heuristique basée sur le modèle en programmation dynamique. L'objectif est de diminuer les temps de calcul sans trop détériorer la qualité des solutions.

Le programme dynamique génère un graphe dont le nombre de nœuds croît exponentiellement avec les paramètres étudiés dans la section 6.5.1 (essentiellement avec le nombre de produits à séquencer). Le principe de l'heuristique proposée est de couper certains des nœuds du graphe pour contrôler la taille du graphe à explorer.

A l'étape 2 de la procédure *Optimal_schedule*, nous vérifions si l'état peut potentiellement améliorer la solution actuelle.

Si $W_{j+1}^k + PLB_{j+1}^k > Retard\ de\ référence$, on n'explore pas l'état. A cette étape nous pouvons couper plus de nœuds en serrant cette inégalité. Notre heuristique propose de ne pas explorer un nœud si $W_{j+1}^k + PLB_{j+1}^k > \boldsymbol{CR} * ReferenceDelay$, CR étant appelé le taux de coupe (CR: $cutting\ rate, CR \in [0,1]$).

Le nombre de nœuds croît jusqu'au milieu du graphe où se trouve le nombre maximum de nœuds par niveau. Ce nombre décroît ensuite aux derniers niveaux du graphe. Le tableau 6.8 présente l'évolution du nombre de nœuds en fonction du niveau pour une des instances testées pour l'étude numérique de complexité (section 6.5.2).

Tableau 6.8 : Evolution du nombre de nœuds en fonction du niveau

Niveau	1	3	5	7	9	11	13	15	17	19	21
Nombre de nœuds	1	25	326	1827	5172	8845	9783	7178	3223	723	55

Nous proposons de contrôler le niveau à partir duquel nous resserrons l'inégalité ci-dessus afin de maitriser le nombre de nœuds coupés. En effet, si on coupe considérablement à partir des premiers niveaux, nous aurons certes un graphe de taille réduite, mais, l'information aux premiers niveaux n'étant pas fiable, ceci peut conduire à un risque élevé de couper des branches comportant la solution optimale ou de bonnes solutions.

Comme le niveau est caractérisé par le nombre de produits restants à séquencer, le niveau choisi sera caractérisé par $\boldsymbol{\lfloor SR * N \rfloor}$, N étant le nombre de produits à séquencer et SR étant le taux caractérisant le début de coupe, un paramètre à choisir (SR: $starting\ rate, SR \in [0,1]$). Si $SR = 0$, on coupe dès le début du graphe et si $SR = 1$, on ne coupe jamais.

Étant donné que plus on s'approche de la fin du graphe, plus la borne inférieure est proche du *Best Partial Delay*, nous avons choisi d'augmenter linéairement le paramètre CR en fonction du pourcentage des produits séquencés jusqu'au dernier niveau où tous les produits seront séquencés. La figure 6.3 montre l'évolution du taux de coupe CR. Nous fixons le taux de début de coupe SR puis nous choisissons une valeur initiale de CR appelée CR_0. A partir de cette valeur, CR augmente linéairement jusqu'au dernier niveau où il prend la valeur 1. Les deux paramètres de l'heuristique sont donc SR et CR_0. Remarquons que quand $SR = 1$ ou $CR_0 = 1$, on retrouve la procédure optimale.

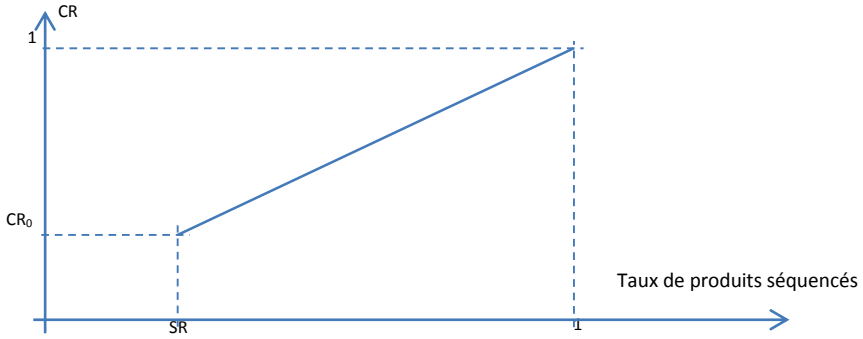


Figure 6.3 Evolution du taux de coupe

6.6.1 Tests académiques

Nous expérimentons la méthode heuristique sur 10 instances. Ces instances sont celles utilisées pour l'analyse expérimentale de complexité de la procédure exacte du programme dynamique pour le cas multi-opérateurs (section 6.5.1). Nous les avons choisies aléatoirement parmi celles dont la solution optimale est connue. Le nombre des opérateurs fluctue entre 4 et 24 opérateurs (tous du type 1). Le nombre de produits varie entre 20 et 24 produits et le nombre de modèles oscille entre 5 et 15 modèles.

Nous faisons varier le paramètre SR dans l'intervalle [0,1 ; 0,9] et CR_0 dans l'intervalle [0,1 ; 0,99]. Nous notons pour chaque couple le temps de calcul et le gap entre la solution trouvée et la solution optimale afin de déterminer les meilleures valeurs de ces paramètres. Nous limitons le temps de calcul à 10 minutes. Les tests avec des valeurs de CR_0 entre 0,1 et 0,7 ne présentent pas de solutions intéressantes. Dans plusieurs cas, les solutions trouvées par la procédure d'initialisation n'ont pas été améliorées pendant le processus d'optimisation. Les résultats pour CR_0 entre 0,8 et 0,99 sont présentés dans le tableau 6.9.

Pour chaque couple (CR_0, SR), nous indiquons tout d'abord la moyenne, pour les 10 instances, du gap entre la solution trouvée et la solution optimale, puis respectivement le

minimum et le maximum du gap. Le gap entre la solution optimale et la solution de l'heuristique se calcule comme suit :

$$Gap = \frac{Solution\ de\ l'heuristique - Solution\ optimale}{Solution\ optimale}$$

Nous indiquons aussi respectivement la moyenne, le minimum et le maximum du gain en temps de calcul par rapport à la procédure de la méthode exacte. Le gain en temps de calcul s'écrit comme suit :

$$Gain = \frac{Temps\ de\ calcul(méth.exacte) - Temps\ de\ calcul\ (heuristique)}{Temps\ de\ calcul(méth.exacte)}$$

Tableau 6.9 : Résultats des tests académiques

SR \ CR_0		**0,8**	**0,9**	**0,95**	**0,97**	**0,99**
0,1	Solution	7,4% [0,7 22,4]%	1% [0 5,7]%	0,4% [0 3,3]%	0,2% [0 1,6]%	0% [0 0]%
	Temps	97,9% [100 93,3]%	80,2% [100 43,3]%	61,1% [100 30]%	41,6% [99,9 26,7]%	9,2% [24,3 -53,3]%
0,2	Solution	6,8% [0,7 15,2]%	1,1% [0 5,7]%	0,4% [0 2,4]%	0,2% [0 1,6]%	0,1% [0 0,8]%
	Temps	98,8% [100 96,9]%	84,2% [100 46,7]%	62,9% [99,9 30]%	43,9% [99,9 26,7]%	13% [27 -25]%
0,3	Solution	7,7% [0,7 17,5]%	1,5% [0 6,8]%	0,4% [0 2,4]%	0,1% [0 0,8]%	0,1% [0 0,8]%
	Temps	96,9% [[100 81,7]%	85,7% [100 50]%	65,8% [100 33,3]%	47,4% [93,3 26,7]%	15% [27 -20]%
0,4	Solution	8,6% [1,6 14,6]%	1,6% [0 6,8]%	0,3% [0 1,6]%	0,1% [0 0,8]%	0,1% [0 0,8]%
	Temps	-5,3% [100 -900]%	48% [100 -301,7]%	42% [99,9 -166,7]%	27,6% [54,1 -133,3]%	9,9% [29,7 -86,7]%
0,5	Solution	8,9% [1,6 16]%	1,9% [0 6,8]%	0,2% [0 1,6]%	0,2% [0 1,6]%	0,1% [0 0,8]%
	Temps	-32,4% [99,3 -900]%	-17,4% [99,3 -900]%	-30,4% [99,5 -900]%	-47,2% [55,6 -900]%	-70,6% [29,7 -900]%
0,6	Solution	10,3% [1,6 19,5]%	2,1% [0,4 6,8]%	0,2% [0 1,6]%	0,2% [0 1,6]%	0,1% [0 0,8]%
	Temps	-220,9% [92,4 -900]%	-38,7% [92,4 -900]%	-38,8% [93,6 -900]%	-50,9% [56,4 -900]%	-70,9% [29,7 -900]%
0,7	Solution	10,8% [1,6 16,5]%	2,4% [0,5 6,8]%	0,6% [0 2,8]%	0,2% [0 1,6]%	0,1% [0 0,8]%
	Temps	-515,1% [72,9 -521,6]%	-84% [72,9 -900]%	-58,3% [62,9 -900]%	-62,6% [61,7 -900]%	-74,1% [24,3 -900]%
0,8	Solution	10,9% [1,6 17,9]%	4,5% [1,6 7,7]%	1,6% [0 7,7]%	0,7% [0 6,5]%	0,6% [0 5,7]%
	Temps	-877,9% [[4,8 -2122,2]%	-262,2% [13,3 -900]%	-92,3% [20 -900]%	-74,9% [25,2 -900]%	-79,1% [20 -900]%
0,9	Solution	12,6% [1,6 19,9]%	4,6% [1,6 6,7]%	1,6% [0 6,5]%	0,8% [0 6,5]%	0% [0 0,3]%
	Temps	-893,2% [-37,6 -2122,2]%	-287,4% [3,3 -900]%	-102,8% [15,4 -900]%	-82,8% [16,7 -900]%	-2,2% 20 [-113,3]%

Nous observons un bon compromis entre le temps de calcul et la qualité de la solution quand CR_0 est dans [0,9 ; 0,95] et SR dans [0,1 ; 0,3] (zone grisée dans le tableau 6.9). Nous atteignons presque la solution optimale (moins de 1,5% de gap en moyenne) tout en gagnant en vitesse d'exécution (temps de calcul réduit de plus de 61%).

Par ailleurs, nous remarquons que quand SR est dans [0,7 ; 0,9] le temps de calcul est plus important que celui de la méthode exacte (valeur négative dans le tableau). Dans ces cas, la limite du temps de calcul est atteinte. Nous commençons à couper tardivement (le taux de début de coupe SR est supérieur à 0,7) mais avec un taux de coupe important (CR_0 est supérieur à 0,8). Ceci peut avoir un effet négatif. En effet, l'heuristique coupe beaucoup de branches qui auraient pu améliorer le $ReferenceDelay$. Or, nous rappelons que $ReferenceDelay$ permet de couper des branches dans la méthode exacte. Si celui-ci est proche de la solution optimale, la procédure coupera efficacement. Nous n'améliorons pas le retard de référence ce qui explique les mauvais résultats dans ces cas.

6.6.2 Tests sur les données du cas d'étude

Dans cette partie nous testons l'heuristique sur les données réelles d'une des deux lignes d'assemblage du centre de montage Volvo à Bourg en Bresse. Nous testons les instances utilisées pour tester la programmation linéaire (voir section 5.4.3). Nous ne prenons en compte que les opérateurs de type 1 (77 opérateurs), puisque la prise en compte des trois types d'opérateurs fait augmenter considérablement la mémoire requise par le programme pour les instances de tailles réelles. Nous testons 9 journées de production d'environ 60 produits chacune en faisant varier les paramètres SR dans l'intervalle [0,1 ; 0,9] et CR_0 dans l'intervalle [0,1 ; 0,99] avec l'objectif de déterminer les meilleurs valeurs de ces paramètres. Nous arrêtons l'optimisation au bout de 10 minutes de calcul. Nous comparons le temps de calcul rapport au programme linéaire (appliqué avec les opérateurs de type 1) arrêté également au bout de 10 minutes de calcul. Nous notons aussi le gap entre la solution trouvée par l'heuristique et celle du programme linéaire.

Dans le tableau 6.10, nous présentons les résultats de cette expérimentation. Pour chaque couple (CR_0, SR), nous indiquons la moyenne, pour les 9 instances, du gap entre la solution trouvée par l'heuristique et celle du programme linéaire. Ce gap se calcule comme suit :

$$Gap = \frac{Solution\ de\ l'heuristique - Solution\ Prog.Linéaire}{Solution\ Prog.Linéaire}$$

Le tableau 6.10 ne présente que les résultats intéressants en termes de qualité de la solution trouvée (CR_0 entre 0,3 et 0,7 et SR entre 0,1 et 0,5). Chaque cellule présente la

moyenne du gap entre la solution trouvée par l'heuristique et celle du programme linéaire pour les 9 journées testées. Le reste des résultats et le minimum et le maximum des gaps sont exposés dans l'annexe D.

Tableau 6.10 : Résultats des tests du cas d'étude

SR \ CR_0	**0,3**	**0,4**	**0,5**	**0,6**	**0,7**
0,1	68,2%	56,7%	54,1%	81,7%	89,0%
0,2	84,9%	63,0%	55,9%	59,0%	85,4%
0,3	97,4%	87,0%	69,9%	63,1%	86,1%
0,4	98,4%	97,6%	90,9%	75,0%	71,3%
0,5	98,4%	98,4%	97,6%	89,9%	77,7%

L'heuristique ne converge pas en moins de 10 minutes pour les instances considérées mais fournit les solutions trouvées au bout de cette limite de temps. Dans tous les tests, la solution trouvée est de moins bonne qualité que celle trouvée par le programme linéaire. Les meilleurs résultats sont trouvés avec CR_0 égale à 0,5 et SR égale à 0,1. Les zones adjacentes à ces valeurs sont aussi intéressantes (voir zone grisée dans le tableau 6.10). En juxtaposant ces résultats avec ceux des tests académiques, nous remarquons que les meilleurs réglages de SR et CR_0 dépendent des instances testées.

L'heuristique proposée permet donc de trouver des solutions intéressantes pour les instances académiques mais pas pour les instances du cas d'étude, et de plus, il semble que les meilleurs réglages de SR et CR_0 dépendent de l'instance testée.

6.7 Conclusion

Dans ce chapitre, nous avons développé un programme dynamique qui permet de modéliser le problème de minimisation des retards cumulés d'une ligne de montage multi-modèles. Il tient compte des caractéristiques du cas industriel (trois types d'opérateurs).

Nous avons testé la procédure optimale sur le cas mono-opérateur puis sur le cas multi-opérateurs. Ces tests ont montré que même s'il est difficile de discerner expérimentalement les facteurs de complexité du modèle proposé, il atteint ses limites au-delà de 24 produits, de 10 modèles et de 32 opérateurs.

Par ailleurs, nous avons développé une procédure heuristique basée sur le modèle en programmation dynamique. Cette heuristique est performante pour les instances académiques en se positionnant sur les bons paramètres. Elle améliore le temps de calcul

tout en donnant des solutions proches de l'optimal. Cependant, l'heuristique est moins performante pour les tests du cas d'étude : elle n'améliore pas les solutions trouvées par le programme linéaire pour des temps de calcul similaires. Par ailleurs, les valeurs des paramètres correspondant aux meilleures performances ne sont pas identiques à celles des tests académiques.

Dans le chapitre suivant, nous allons utiliser des métaheuristiques dans le but de diminuer le temps de calcul tout en gardant la qualité des solutions afin de pouvoir l'appliquer dans le contexte industriel.

Chapitre 7 : Résolution du problème de séquencement par des métaheuristiques

7.1 Introduction

Dans les deux chapitres précédents nous avons proposé deux méthodes exactes pour la résolution du problème de minimisation des retards cumulés d'une ligne de montage multi-modèles. L'apport de ces méthodes est la modélisation du problème. Cependant, les temps de calcul sont longs pour la résolution exacte des instances industrielles. Une heuristique basée sur la programmation dynamique a été proposée dans le chapitre précédent mais elle n'est pas très satisfaisante en termes de qualité des résultats obtenus. Dans ce chapitre, nous continuons à explorer les méthodes approchées pour la résolution du problème en développant des métaheuristiques. Nous proposons deux métaheuristiques : les algorithmes génétiques et le recuit simulé. Nous combinons ensuite les deux méthodes pour profiter des points de forts chacune d'entre elles. L'ensemble des trois méthodes est testé sur différentes instances mono-opérateur et multi-opérateurs, académiques et issues du cas d'étude. Ce travail a été en partie réalisée dans le cadre de l'encadrement du stage de master de Monsieur Abduh-Sayid ALBANA et publié dans (Albana et al., 2014).

7.2 Les algorithmes génétiques

Les algorithmes génétiques (AG) sont des algorithmes d'optimisation fondés sur les mécanismes de la sélection naturelle et de la génétique, ils ont été initiés par John Holland (Holland, 1975). Ils se basent sur trois éléments : la reproduction, le croisement et la mutation.

Le principe est de partir d'une population de solutions (appelées chromosomes ou individus) initiales qu'on évalue (grâce à une fonction de fitness). Sur la base de ces évaluations, on crée une nouvelle population de solutions potentielles en utilisant des opérations évolutionnaires. Quelques solutions se reproduisent, d'autres disparaissent et seules les solutions les mieux adaptées sont présumées survivre. On répète ces opérations évolutionnaires jusqu'à ce qu'on trouve une solution satisfaisante (Goldberg et Holland, 1988).

7.2.1 Le codage des données

Cette étape consiste à définir et coder le problème. Nous allons donc procéder à la représentation d'une solution par une structure de données spécifiques. Le codage de

chaque chromosome est primordial dans la conception d'un algorithme génétique dont dépend notamment l'utilisation des opérateurs de transformations (Yang, 2010).

Une solution (chromosome) est une séquence de gènes. Dans notre cas un chromosome est une séquence de produits et chaque gène représente un produit *i* dans une position *j* (figure 7.1).

	Position 1	Position 2	Position 3	Position 4	Position 5
Séquence:	3	1	5	4	2
	↑ Gène	↑ Gène	↑ Gène	↑ Gène	↑ Gène

Figure 7.1 : Représentation d'un chromosome et des gènes

7.2.2 L'évaluation des solutions

Cette étape permet d'évaluer la performance des individus et ainsi les comparer avec les autres. Cette évaluation est établie grâce à une fonction d'adaptation appelée aussi « *Fitness* ». Pour les problèmes de maximisation, la fonction d'adaptation est égale à une fonction linéaire de la fonction objectif *f(x)*. Pour les problèmes de minimisation, elle est l'inverse de *f(x)*. Nous allons utiliser la fonction d'adaptation donnée dans (Santosa et Willy, 2011) :

$$F(x) = \frac{1}{1 + f(x)} = \frac{1}{1 + \sum_{p=1}^{P} \sum_{j=1}^{N} w_{jp}} \qquad (7.1)$$

Avec : $F(x)$: la fonction d'adaptation

$f(x)$: la fonction objectif

w_{jp} : retard de l'opérateur p pour le produit de la position j

7.2.3 L'élitisme

La sélection permet d'identifier les meilleurs individus d'une population et d'éliminer les mauvais. Ceci est basé sur l'évaluation de la performance de chaque individu grâce à la fonction d'adaptation. L'élitisme est une méthode de sélection qui permet de sélectionner l'individu le plus performant et d'en faire des copies afin de remplacer les individus les moins performants de la population. Dans notre cas, nous copions l'individu le plus performant trois fois pour remplacer les trois individus les moins performants.

7.2.4 L'opérateur de croisement

L'opérateur de croisement est appliqué à deux parents pour donner naissance à deux enfants en espérant qu'ils hériteront des bons gènes de chacun des parents. La figure 7.2

présente un exemple de trois croisements de 6 chromosomes. Chaque croisement est appliqué à deux parents choisis aléatoirement parmi les 6 chromosomes de départ. Un chromosome ne peut pas être croisé avec lui-même et ne peut pas être tiré deux fois. Chaque croisement donne naissance à deux enfants qu'on garde dans la population seulement si leurs fonctions d'adaptation sont meilleures que celles de leurs parents. Dans ce cas, le ou les mauvais parents sont supprimés de la population.

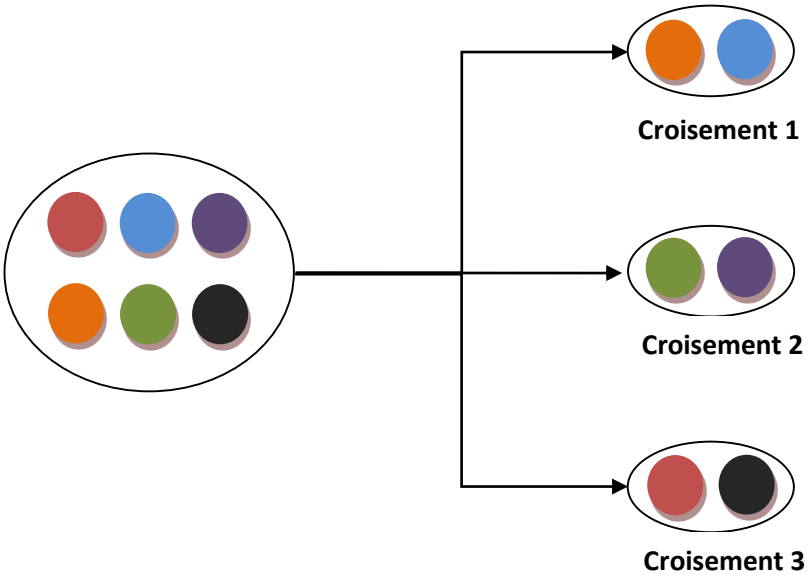


Figure 7.2 : Sélection des parents pour le croisement

On utilise le croisement à un point. Ceci consiste à choisir aléatoirement une position dans la séquence. Toutes les données des parents au-delà de cette position seront inter-changées pour former deux enfants (voir l'exemple de la figure 7.3).

Dans la figure 7.3, nous croisons deux parents A et B en utilisant un croisement à partir de la deuxième position. Suite à l'opération de croisement, nous obtenons deux enfants. L'enfant 1 est le résultat de la combinaison des deux premiers gènes du parent A et des trois derniers gènes du parent B. L'enfant 2 est né de la combinaison inverse.

Aucun des deux enfants n'est valide : le même produit (gène) est répété deux fois dans la séquence et un des produits manque à chaque enfant. Nous devons alors corriger ces chromosomes. La correction s'applique à la partie avant le point de croisement. Dans la première partie (avant le point de croisement) de l'enfant 1, nous mettons les produits manquants à la deuxième partie. Nous les positionnons selon leur ordre d'apparition dans le parent A. Par exemple, à la deuxième partie de l'enfant 1, il manque les produits 2 et 4. Leur ordre d'apparition dans le parent A est 2-4. On remplace les deux gènes avant le point de croisement par 2-4. On fait de même pour l'enfant 2. A la deuxième partie de

l'enfant 2, il manque les produits 2 et 5. Leur ordre d'apparition dans le parent B est 2-5. On remplace alors les deux gènes avant le point de croisement par 2-5.

Quand les deux enfants sont « sains » c'est-à-dire réalisables, nous calculons leur fonction d'adaptation pour les évaluer. Si l'enfant est meilleur que le parent correspondant (on choisit que l'enfant 1 correspond au parent A et l'enfant 2 correspond au parent B), on l'ajoute à la population pour remplacer le parent correspondant. Dans le cas contraire, les parents restent dans la population. Ceci permet de garder les meilleurs individus dans la population.

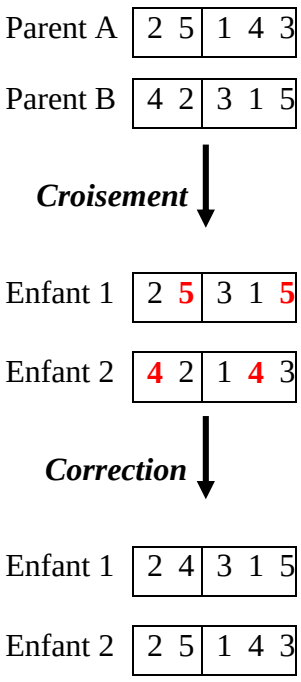


Figure 7.3 : Exemple d'une opération de croisement

7.2.5 L'opérateur de mutation

L'opérateur de mutation permet de garder de la diversité dans la population. Trois types de mutation sont effectués dans notre cas : le remplacement, l'inversion et le glissement. Ces mutations sont décrites dans la figure 7.4. Le choix des positions de remplacement, d'inversion ou de glissement est fait aléatoirement.

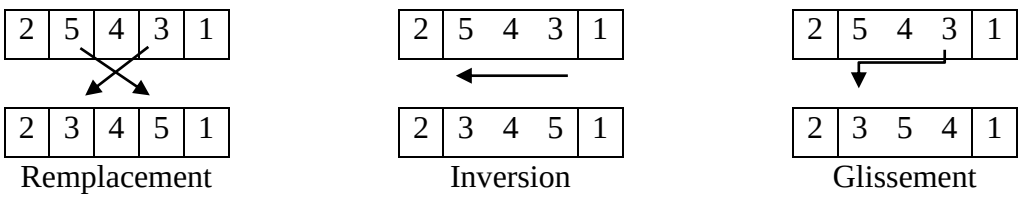


Figure 7.4 : Trois opérations de mutation

Pour l'opération de remplacement de la figure 7.4, le choix aléatoire des positions donne 2 et 4. Nous échangeons les gènes de ces positions. Le deuxième exemple est une inversion de la sous séquence entre les positions 2 et 4. On remplace alors 5-4-3 par 3-4-5. L'opération de glissement est illustrée dans le dernier exemple. Le choix aléatoire des positions donne 4 et 2. On met le gène de la position 4 à la position 2 et on glisse le reste.

7.2.6 Fonctionnement de l'algorithme

Nous générons tout d'abord la population initiale. Ceci consiste à choisir des individus de départ que l'algorithme va faire évoluer. Dans notre cas, ces individus sont générés aléatoirement : les valeurs des gènes sont tirées au hasard selon une distribution uniforme tout en réalisant des séquences réalisables (pas deux fois le même produit).

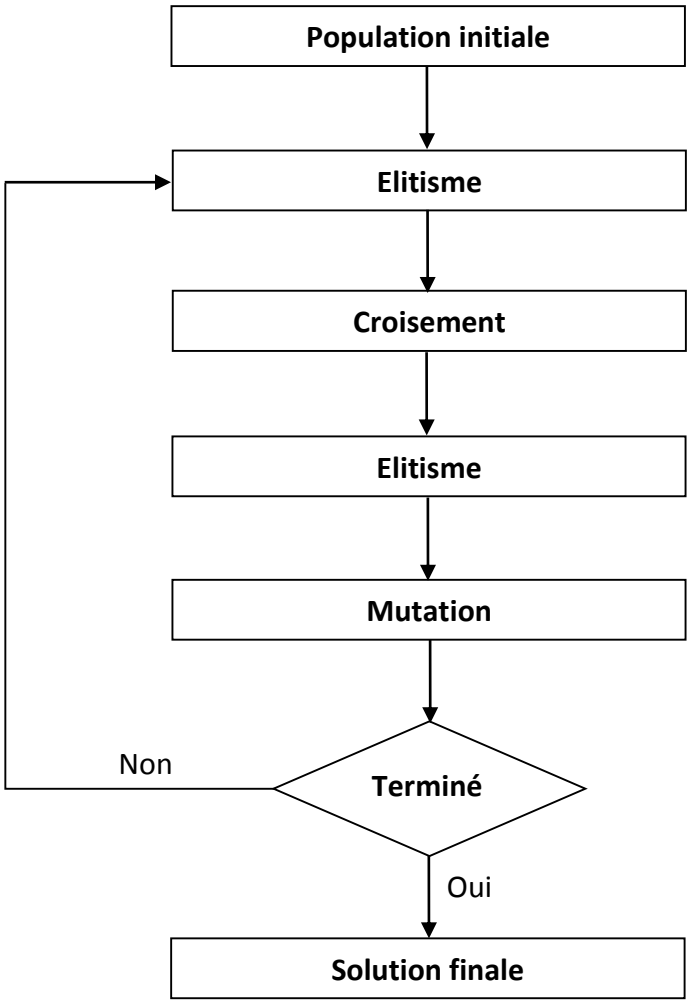


Figure 7.5 : Fonctionnement de l'algorithme génétique

Dans les algorithmes génétiques, plusieurs paramètres sont à définir : le nombre d'individus dans la population, le nombre de chromosomes issus de l'élitisme, la probabilité de procéder à un croissement et la probabilité de procéder à une mutation.

Dans notre algorithme, on a pris 10 individus dans la population. Les probabilités de procéder à un croisement et à une mutation sont égales à 1. Quand on procède à un élitisme, le meilleur individu est copié trois fois. La mutation (remplacement, inversion et glissement) est effectuée sur ces trois copies (une mutation pour chaque individu). La figure 7.5 explique le fonctionnement de l'algorithme génétique.

```
Algorithme génétique

Générer la population initiale (10 individus)

Tant que (tous les individus n'ont pas la même fitness) refaire

  Appliquer Elitisme (3 copies du meilleur individu)

  Croisement
  Pour i de 1 à 3 faire

  Sélectionner deux parents parmi les 6 individus restants
  Appliquer Croisement

   Pour chaque enfant
     Si l'enfant est meilleur que le parent correspondant
      Remplacer le parent
     Sinon
      Annuler le croisement
     Fin si
   Fin pour

  Fin pour

  Appliquer Elitisme (3 copies du meilleur individu)

  Mutation

  Sélectionner un des 3 individus issus de l'élitisme
  Appliquer Mutation 1 : remplacement

  Sélectionner un des 2 individus restants
  Appliquer Mutation 2 : inversion

  Appliquer Mutation 3 à l'individu restant: glissement

Fin
```

Figure 7.6 : Pseudo code de l'algorithme génétique

Le pseudo code de notre algorithme génétique est présenté dans la figure 7.6. Nous appliquons un élitisme à la population initiale de 10 individus en copiant le meilleur individu trois fois pour remplacer les trois plus mauvaises solutions. Nous procédons à trois croissements pour les 6 individus restants (en s'assurant que chaque individu est utilisé une et une seule fois). Ensuite, un deuxième élitisme est effectué. Puis, les trois opérations de mutation sont appliquées aux trois individus copiés lors de l'élitisme afin de garder de la diversité dans la population. Un cycle d'évolution complet est formé par l'application des opérateurs de sélection, croisement et mutation sur une population d'individus. L'algorithme s'arrête quand tous les individus ont la même fitness.

7.2.7 Résultats numériques

Nous expérimentons l'algorithme génétique proposé pour le cas mono-opérateur et multi-opérateurs sur des instances académiques et d'autres issues du cas d'étude.

7.2.7.1 Cas mono-opérateur de type 1

Cette étude expérimentale permet de tester l'algorithme génétique. Elle est constituée de 17 instances d'un seul opérateur chacune et de 20 produits à séquencer. Pour chaque instance, dix tests sont effectués (en partant de populations initiales différentes). Le tableau 7.1 présente le meilleur résultat et le pire résultat en termes de qualité de la solution et de temps de calcul. La meilleure solution ne correspond pas forcément au meilleur temps de calcul. Les résultats sont comparés aux résultats du programme linéaire.

Le gap entre la solution trouvée par l'heuristique et celle du programme linéaire est calculé grâce à l'équation (7.2) :

$$Gap = \frac{Solution\ de\ l'algo.\ génétique - Solution\ du\ prog.\ linéaire}{Solution\ du\ prog.\ linéaire} \quad (7.2)$$

Les cellules surlignées dans le tableau indiquent que soit le temps de calcul est plus long en utilisant l'algorithme génétique, soit la solution optimale n'est pas atteinte par au moins un des 10 tests effectués. Le gap prend une valeur négative si l'heuristique trouve une meilleure solution que le programme linéaire (ce qui peut se produire dans les cas où le programme linéaire n'atteint pas la solution optimale).

L'algorithme génétique ne trouve pas la solution optimale pour tous les tests. Par exemple, pour l'instance TH_MONO1_8, deux des 10 tests effectués atteignent seulement 98% de l'optimal. Par contre, pour les 17 instances testées la solution optimale est obtenue par au moins un des 10 tests. Nous remarquons que pour ces instances de petite taille l'algorithme génétique requiert plus de temps de calcul que le programme linéaire.

Nous testons par ailleurs des instances mono-opérateur (43 produits) que le programme linéaire n'arrive pas à résoudre au bout de 30 minutes. Le tableau 7.2 présente la meilleure solution trouvée par le programme linéaire et les résultats de l'algorithme génétique.

Tableau 7.1 : Résultats des tests des instances académiques mono-opérateur 1

Instance	Programme linéaire		Algorithme génétique					
	Solution optimale	Temps de calcul (s)	Objectif		Temps de calcul (s)		Gap	
			Pire	Meilleur	Plus long	Plus court	Pire	Meilleur
TH_MONO1_1	3,4	10,00	3,4	3,4	16,40	2,52	0%	0%
TH_MONO1_2	2,1	4,70	2,1	2,1	6,74	2,05	0%	0%
TH_MONO1_3	5,2	15,00	5,2	5,2	18,74	2,48	0%	0%
TH_MONO1_4	2,6	97,00	2,8	2,6	16,77	3,24	8%	0%
TH_MONO1_5	2,8	2,30	2,8	2,8	5,97	1,15	0%	0%
TH_MONO1_6	0,2	0,30	0,2	0,2	3,10	0,55	0%	0%
TH_MONO1_7	4,7	316,00	4,7	4,7	9,54	3,32	0%	0%
TH_MONO1_8	5,0	6,40	5,1	5,0	29,33	9,28	2%	0%
TH_MONO1_9	5,8	0,90	6,1	5,8	20,41	7,77	5%	0%
TH_MONO1_10	3,0	504,00	3,0	3,0	7,88	1,78	0%	0%
TH_MONO1_11	2,4	69,00	2,4	2,4	7,48	2,32	0%	0%
TH_MONO1_12	2,8	2,00	2,8	2,8	2,53	1,22	0%	0%
TH_MONO1_13	2,0	1,00	2,0	2,0	5,26	0,95	0%	0%
TH_MONO1_14	2,9	2,00	2,9	2,9	10,18	2,34	0%	0%
TH_MONO1_15	2,2	2,20	2,2	2,2	1,55	0,77	0%	0%
TH_MONO1_16	6,1	3,00	6,1	6,1	5,52	3,19	0%	0%
TH_MONO1_17	5,7	470,00	5,8	5,7	7,98	4,51	2%	0%

Le tableau 7.2 montre que l'algorithme génétique offre un très bon compromis entre la qualité de la solution et le temps de calcul pour les instances testées. En effet, pour chacune de ces trois instances, on obtient, par au moins un des 10 tests, une solution identique à celle obtenue par le programme linéaire (en 1800 secondes) avec des temps beaucoup plus faibles.

Tableau 7.2 : Résultats des tests des instances académiques mono-opérateur 2

Instance	Programme linéaire		Algorithme génétique					
	Solution obtenue	Temps de calcul (s)	Objectif		Temps de calcul (s)		Gap	
			Pire	Meilleur	Plus long	Plus court	Pire	Meilleur
TH_MONO2_1	6,0	1800,00	6,0	6,0	36,32	7,91	0%	0%
TH_MONO2_2	8,4	1800,00	8,6	8,4	56,68	17,73	2%	0%
TH_MONO2_3	6,3	1800,00	6,5	6,3	30,09	5,24	3%	0%

L'étude expérimentale pour le cas mono-opérateur a montré que l'algorithme génétique donne des performances similaires au programme linéaire en un temps de calcul plus long pour certaines instances, et en un temps de calcul plus court pour d'autres. Nous allons tester dans la prochaine section la performance de l'algorithme génétique pour le cas multi-opérateurs.

7.2.7.2 Cas multi-opérateurs de type 1

Nous utilisons tous d'abord des instances issues de la littérature (Bautista et Cano, 2008). Ces instances ont déjà été testées sur le programme linéaire. Nous les expérimentons à l'aide de l'algorithme génétique. Les résultats sont donnés dans l'annexe B. Les résultats montrent tout d'abord que, le programme linéaire converge plus rapidement que l'algorithme génétique mais les temps de calcul de l'algorithme génétique restent très raisonnables et ne dépassent que rarement les 30 secondes. On observe aussi que l'algorithme génétique trouve des solutions optimales ou proches de l'optimale.

Par ailleurs, nous expérimentons d'autres instances académiques générées aléatoirement de 20 produits et de 4 opérateurs. Les résultats sont exposés dans le tableau 7.3. Les résultats montrent que pour ces instances l'algorithme génétique ne trouve pas toujours la solution optimale contrairement au programme linéaire mais pour chacune des trois instances, la solution optimale a été trouvée par au moins un des 10 tests. L'algorithme génétique trouve des solutions proches de l'optimale et parfois optimales dans un temps de calcul acceptable. L'algorithme génétique est intéressant dans ces cas : le temps de calcul est nettement moins élevé que celui du programme linéaire sans une perte importante de performance.

Tableau 7.3 : Résultats des tests des instances académiques multi-opérateur (type 1)

Instance	Programme linéaire		Algorithme génétique					
	Solution optimale	Temps de calcul (s)	Objectif		Temps de calcul (s)		Gap	
			Pire	Meilleur	Plus long	Plus court	Pire	Meilleur
TH_MULTI_1	23,6	1373,00	23,8	23,6	20,25	6,40	1%	0%
TH_MULTI_2	16,0	136,00	17,5	16,0	23,62	5,69	9%	0%
TH_MULTI_3	5,5	311,00	5,6	5,5	39,58	5,44	2%	0%

7.2.7.3 Prise en compte des opérateurs de type 2 et 3

- **Instances académiques**

Nous avons pris en compte les opérateurs de types 2 et 3 dans l'algorithme génétique. Nous testons l'algorithme génétique sur les instances académiques déjà utilisées pour tester le programme linéaire et le programme dynamique (tableaux 5.11, 5.12 et 5.13).

Tableau 7.4 : Résultats des tests académiques avec des opérateurs de type 2 et 3

Instances	Programme linéaire		Programme dynamique	Algorithme génétique					
	Solution optimale	Temps de calcul (s)	Temps de calcul (s)	Objectif		Temps de calcul (s)		Gap	
				Pire	Meilleur	Plus long	Plus court	Pire	Meilleur
1	6,02	0,60	0,14	6,65	6,02	5,34	4,68	10%	0%
2	10,20	0,30	0,11	11,20	10,20	5,26	3,35	10%	0%
3	19,46	1,76	151	22,25	20,54	18,96	13,46	14%	6%

Les résultats, donnés dans le tableau 7.4, montrent que l'algorithme génétique atteint la solution optimale dans le meilleur des cas pour les deux premières instances. Les solutions trouvées dans le pire cas sont bonnes. Le temps de calcul est plus long que celui de la programmation linéaire pour ces instances mais ne dépasse pas les 20 secondes. L'algorithme génétique n'est donc pas aussi performant que le programme linéaire qui garantit l'optimal en un temps de calcul très court pour ces instances de petites tailles.

- **Instances du cas d'étude**

Dans cette partie, nous utilisons l'algorithme génétique pour résoudre le problème de séquencement de journées réelles issues de notre cas d'étude. Les instances testées correspondent à des journées de production d'une ligne d'assemblage du centre de montage Volvo à Bourg en Bresse. Sur cette ligne nous avons 92 opérateurs : 77 opérateurs de type 1, 12 opérateurs de type 2 et 3 opérateurs de type 3. Pour les journées étudiées, les durées des opérations affectées aux opérateurs de type 3 sont les mêmes pour tous les véhicules. Nous ne les prendrons donc pas en compte dans nos tests puisque la répartition de la charge de ces opérateurs sera la même quel que soit l'ordre de passage des véhicules sur la ligne. Les instances testées correspondent alors à 89 opérateurs. Les instances ont été testées auparavant en utilisant le programme linéaire (chapitre 5, section 5.4.3) : l'optimisation est arrêtée au bout de trois heures de calcul et aucun des tests ne converge dans cette limite de temps. Nous comparons ces résultats à ceux de l'algorithme génétique. Les résultats sont donnés dans le tableau 7.5.

D'après le tableau 7.5, en un temps beaucoup moins important (entre 6 et 11 minutes vis-à-vis des trois heures pour le programme linéaire), l'algorithme génétique trouve des solutions proches de celles trouvées par le programme linéaire et les améliore même très régulièrement (jusqu'à 5% du résultat du programme linéaire). L'algorithme génétique améliore de 35,7% en moyenne la fonction objectif des séquences obtenues par rapport aux séquences de la procédure actuellement utilisée dans le cas d'étude.

Tableau 7.5 : Résultats des tests des instances du cas d'étude

Instance	Programme linéaire		Algorithme génétique					
	Solution obtenue	Temps de calcul (s)	Objectif		Temps de calcul (s)		Gap	
			Pire	Meilleur	Plus long	Plus court	Pire	Meilleur
1	800,8	10800,00	865,0	794,0	560,89	424,39	8%	-1%
2	585,1	10800,00	593,0	573,1	485,23	429,92	1%	-2%
3	763,4	10800,00	782,5	769,4	628,78	447,01	3%	1%
4	914,5	10800,00	944,9	896,9	560,84	359,70	3%	-2%
5	1823,4	10800,00	1801,6	1736,0	551,91	468,33	-1%	-5%
6	804,4	10800,00	875,2	801,0	557,28	428,24	9%	0%
7	814,8	10800,00	828,4	802,6	597,65	399,40	2%	-2%
8	1450,0	10800,00	1512,7	1426,8	581,64	423,53	4%	-2%
9	800,3	10800,00	793,0	765,7	592,07	420,41	-1%	-4%

En conclusion, la qualité des solutions trouvées par l'algorithme génétique proposé est similaire à la qualité de celles trouvées par le programme linéaire pour les instances de petites tailles (cas mono-opérateur et instances académiques du cas multi-opérateurs) mais pour la majorité de ces instances, les temps de calcul de l'algorithme génétique sont plus élevés que ceux du programme linéaire tout en restant très raisonnables. Pour les instances du cas industriel, l'algorithme génétique trouve généralement des solutions meilleures que celles trouvées au bout de trois heures de calcul par le programme linéaire alors que les temps de calcul de l'algorithme génétique ne dépassent pas les 11 minutes Les performances de cet algorithme sont donc très intéressantes dans un objectif d'application industrielle. Dans la prochaine section, nous allons développer une autre métaheuristique en espérant résoudre les instances du cas d'étude en un temps de calcul moins élevé que celui de l'algorithme génétique.

7.3 Le recuit simulé

Le recuit simulé a été introduit par (Kirkpatrick et al., 1983). Cette métaheuristique est basée sur une analogie à une technique utilisée par les métallurgistes pour obtenir un alliage sans défaut. Le recuit simulé utilise une stratégie pour éviter les optima locaux en acceptant parfois une solution plus mauvaise que la meilleure solution courante.

7.3.1 Fonctionnement de l'algorithme

Le principe est de partir d'une solution initiale S_0 et d'une température initiale T_0. A chaque itération, nous générons une solution voisine de la solution actuelle. Dans notre cas, nous utilisons une mutation : il s'agit de l'inversion expliquée dans la section

précédente. La solution trouvée S' est évaluée grâce à sa valeur de l'objectif. Si la solution est meilleure que la solution actuelle S, nous la gardons pour la prochaine itération. Dans le cas contraire, la solution S' peut être acceptée même si elle détériore l'objectif. La probabilité d'acceptation est définie par la distribution de Boltzmann suivante :

$$P(E,T) = exp^{-\frac{\Delta E}{T}}$$

Avec :

ΔE : différence entre les valeurs de la fonction objectif des solutions S et S'.

T : température actuelle

Nous générons un nombre aléatoire. Si ce nombre est inférieure à *P(E,T)*, la solution S' est acceptée. À chaque cycle, la température décroit grâce à un paramètre de refroidissement *c*. Plus T est petite (c'est-à-dire plus le nombre de cycles est important), moins on accepte une dégradation de la solution. Le pseudo code de notre algorithme de recuit simulé est présenté dans la figure 7.7.

```
Recuit simulé
Paramètres :
T0 : Température initiale
c : paramètre de refroidissement

Générer une solution initiale : S ← S0
Initialiser la température : T ← T0
Tant que (Nombre de cycles < 10) refaire
    Tant que (Nombre d'itérations < 100) refaire
    Générer une nouvelle séquence S' voisine de S
    Si S' est meilleure que S alors
      S ← S'
    Sinon
      Delta ← objectif (S') – objectif (S)
      Probabilité ← nombre aléatoire entre 0 et 1
      Probabilité d'acceptation ← exp(-Delta/T)
      Si Probabilité <= Probabilité d'acceptation alors
        S ← S'
      Fin si
    Fin si
    Itération ← Itération + 1
    Fin
Mettre à jour la température : T ← T * c
Cycle ← Cycle +1
Fin
```

Figure 7.7 : Pseudo code du recuit simulé

La performance de cette méthode dépend des paramètres choisis. Nous expérimentons alors quelques tests pour choisir ces paramètres. Pour ceci, nous testons la première instance du cas d'étude (en ne considérant que les opérateurs de type 1) avec différents paramètres comme indiqué dans le tableau 7.6.

Tableau 7.6 : Paramètres du recuit simulé

Température initiale $\mathbf{T_0}$	Paramètre de refroidissement **c**
1 000 000	0,8
100 000	0,5
10 000	0,2

En faisant varier les valeurs de ces paramètres, nous avons neuf combinaisons possibles dont les résultats sont présentés dans le tableau 7.7.

Chaque ligne représente la moyenne de dix tests. Les solutions du recuit simulé sont trouvées au bout d'environ 17 secondes. Nous remarquons que la performance du recuit simulé dépend des valeurs des paramètres. Pour le reste des expérimentations, nous choisissons la troisième configuration puisque elle a la meilleure solution dans le pire cas. De plus, cette configuration conduit à la plus petite variation en termes de qualité entre les solutions des 10 tests (le plus petit écart entre la pire solution et la meilleure solution).

Tableau 7.7 : Résultats des variations des paramètres

Combinaison	Température initiale T_0	Paramètre de refroidissement c	Recuit simulé			
			Objectif		Gap	
			Pire	Meilleur	Pire	Meilleur
1	1 000 000	0,8	815,5	674,2	21%	0%
2	100 000	0,8	794,3	698,8	18%	3%
3	10 000	0,8	757,7	679,2	12%	1%
4	1 000 000	0,5	814,9	686,9	21%	2%
5	100 000	0,5	852,0	682,4	26%	1%
6	10 000	0,5	821,1	715,0	22%	6%
7	1 000 000	0,2	838,6	699,3	24%	4%
8	100 000	0,2	805,2	669,0	19%	-1%
9	10 000	0,2	838,9	689,0	24%	2%

Dans la prochaine section, nous testons le recuit simulé sur les instances qui ont déjà servi à la validation de l'algorithme génétique afin de comparer les deux métaheuristiques.

7.3.2 Résultats numériques

Nous expérimentons l'algorithme de recuit simulé proposé pour le cas mono-opérateur et multi-opérateurs sur des instances académiques et d'autres issues du cas d'étude.

7.3.2.1 Cas mono-opérateur de type 1

Nous testons les instances académiques déjà utilisées pour valider l'algorithme génétique. En premier lieu nous expérimentons 17 instances (cf. tableau 7.1) d'un seul opérateur chacune et de 20 produits à séquencer (tableau 7.8).

Tableau 7.8 : Résultats des tests des instances académiques mono-opérateur 1

Instance	Programme linéaire		Recuit simulé					
	Solution optimale	Temps de calcul (s)	Objectif		Temps de calcul (s)		Gap	
			Pire	Meilleur	Plus long	Plus court	Pire	Meilleur
TH_MONO1_1	3,4	10,00	3,4	3,4	2,27	2,23	0%	0%
TH_MONO1_2	2,1	4,70	2,1	2,1	2,25	2,23	0%	0%
TH_MONO1_3	5,2	15,00	5,2	5,2	2,27	2,25	0%	0%
TH_MONO1_4	2,6	97,00	2,6	2,6	2,27	2,25	0%	0%
TH_MONO1_5	2,8	2,30	2,8	2,8	2,25	2,23	0%	0%
TH_MONO1_6	0,2	0,30	0,2	0,2	2,24	2,22	0%	0%
TH_MONO1_7	4,7	316,00	4,7	4,7	2,26	2,24	0%	0%
TH_MONO1_8	5,0	6,40	5,0	5,0	2,27	2,25	0%	0%
TH_MONO1_9	5,8	0,90	6,2	5,8	2,28	2,25	7%	0%
TH_MONO1_10	3,0	504,00	3,0	3,0	2,26	2,24	0%	0%
TH_MONO1_11	2,4	69,00	2,4	2,4	2,24	2,23	0%	0%
TH_MONO1_12	2,8	2,00	2,8	2,8	2,23	2,22	0%	0%
TH_MONO1_13	2,0	1,00	2,0	2,0	2,23	2,22	0%	0%
TH_MONO1_14	2,9	2,00	2,9	2,9	2,24	2,22	0%	0%
TH_MONO1_15	2,2	2,20	2,2	2,2	2,24	2,22	0%	0%
TH_MONO1_16	6,1	3,00	6,1	6,1	2,24	2,23	0%	0%
TH_MONO1_17	5,7	470,00	6,3	5,7	2,24	2,23	11%	0%

Le recuit simulé trouve toujours la solution optimale sur les 10 tests pour 15 instances (sur les 17 instances testées). Il est plus rapide que l'algorithme génétique mais parfois plus long que la programmation linéaire.

En deuxième lieu, nous utilisons les instances académiques à 43 produits déjà testées sur l'algorithme génétique (cf. tableau 7.2). Nous rappelons que le programme linéaire

n'arrive pas à trouver la solution optimale (ou à prouver son optimalité) en 30 minutes. Les résultats sont donnés dans le tableau 7.9.

Tableau 7.9 : Résultats des tests des instances académiques mono-opérateur 2

Instance	Programme linéaire		Recuit simulé					
	Solution obtenue	Temps de calcul (s)	Objectif		Temps de calcul (s)		Gap	
			Pire	Meilleur	Plus long	Plus court	Pire	Meilleur
TH_MONO2_1	6,0	1800,00	6,0	6,0	2,06	2,04	0%	0%
TH_MONO2_2	8,4	1800,00	8,4	8,4	2,06	2,04	0%	0%
TH_MONO2_3	6,3	1800,00	6,3	6,3	2,06	2,04	0%	0%

Le tableau 7.9 montre qu'au bout de deux secondes, le recuit simulé trouve toujours la solution trouvée au bout de 30 minutes de calcul par le programme linéaire.

7.3.2.2 Cas multi-opérateurs de type 1

Nous testons les instances académiques générées aléatoirement pour valider l'algorithme génétique (cf. tableau 7.3). Les résultats sont exposés dans le tableau 7.10.

Tableau 7.10 : Résultats des tests des instances académiques multi-opérateurs (type 1)

Instance	Programme linéaire		Recuit simulé					
	Solution optimale	Temps de calcul (s)	Objectif		Temps de calcul (s)		Gap	
			Pire	Meilleur	Plus long	Plus court	Pire	Meilleur
TH_MULTI_1	23,6	1373,00	23,6	23,6	2,48	2,46	0%	0%
TH_MULTI_2	16,0	136,00	16,7	16,2	2,48	2,47	4%	1%
TH_MULTI_3	5,5	311,00	5,5	5,5	2,46	2,45	0%	0%

Le recuit simulé trouve dans tous les cas la solution optimale pour deux instances sur les trois testées. Le temps de calcul est moins important que pour le programme linéaire et l'algorithme génétique. Par ailleurs, l'écart type entre les 10 solutions trouvées pour chaque instance est moins important que pour celui de l'algorithme génétique.

7.3.2.3 Prise en compte des opérateurs de type 2 et 3

- **Instances académiques :**

Nous testons le recuit simulé sur les instances académiques utilisées pour tester l'algorithme génétique (cf. tableau 7.4). Les résultats sont donnés dans le tableau 7.11.

Les résultats montrent que les temps de calcul du recuit simulé sont similaires à ceux de l'algorithme génétique pour ces instances. La qualité des solutions du recuit simulé pour les pires cas est moins bonne que celle de l'algorithme génétique pour les deux

premières instances alors que dans les meilleurs cas le recuit simulé obtient, comme l'algorithme génétique, la solution optimale. Pour la troisième instance, le recuit simulé est un peu meilleur que l'algorithme génétique dans le meilleur cas. Le programme linéaire reste plus performant que le recuit simulé pour ces instances de petites tailles.

Tableau 7.11 : Résultats des tests académiques avec des opérateurs de type 2 et 3

Instance	Programme linéaire		Recuit simulé					
	Solution optimale	Temps de calcul (s)	Objectif		Temps de calcul (s)		Gap	
			Pire	Meilleur	Plus long	Plus court	Pire	Meilleur
1	6,02	0,60	7,23	6,02	6,66	6,27	20%	0%
2	10,20	0,30	11,90	10,20	6,32	6,06	17%	0%
3	19,46	1,76	22,60	20,21	11,72	10,95	16%	4%

- **Instances du cas d'étude**

Nous expérimentons le recuit simulé sur les instances du cas d'étude (cf. tableau 7.5). Les résultats sont donnés dans le tableau 7.12. Nous rappelons que l'optimisation est arrêtée au bout de trois heures pour la programmation linéaire et qu'aucun test ne converge dans cette limite de temps.

Le temps de calcul du recuit simulé est nettement moins important que celui du programme linéaire et de l'algorithme génétique. Par contre, la performance de l'algorithme génétique reste meilleure que celle du recuit simulé qui atteint dans le pire des cas 78% du résultat du programme linéaire et dans le meilleur des cas 99,7%.

Tableau 7.12 : Résultats des tests des instances du cas d'étude

Instance	Programme linéaire		Recuit simulé					
	Solution obtenue	Temps de calcul (s)	Objectif		Temps de calcul (s)		Gap	
			Pire	Meilleur	Plus long	Plus court	Pire	Meilleur
1	800,8	10800,00	976,1	863,5	75,86	75,45	22%	8%
2	585,1	10800,00	707,5	618,4	75,66	75,44	21%	6%
3	763,4	10800,00	872,8	791,3	76,20	75,92	14%	4%
4	914,5	10800,00	1105,2	973,8	73,99	73,80	21%	6%
5	1823,4	10800,00	2021,3	1894,8	78,31	75,36	11%	4%
6	804,4	10800,00	963,6	834,6	78,65	76,08	20%	4%
7	814,8	10800,00	872,0	817,3	79,56	77,04	7%	0%
8	1450,0	10800,00	1658,6	1538,0	76,39	75,86	14%	6%
9	800,3	10800,00	850,0	812,9	78,52	75,85	6%	2%

En conclusion, la qualité des solutions trouvées par le recuit simulé proposé est un peu meilleure que la qualité de celles trouvé par l'algorithme génétique pour les instances de

petites tailles (cas mono-opérateur et instances académiques du cas multi-opérateurs). Ces solutions sont trouvées en des temps de calcul nettement inférieurs à ceux de l'algorithme génétique. Pour les instances du cas industriel, le recuit simulé reste très rapide (moins de 80 secondes de temps de calcul) mais la qualité des solutions est inférieure à celles des solutions obtenues par l'algorithme génétique. Ceci reste acceptable puisque, au pire des cas, le recuit simulé obtient, en moins de 80 secondes, une solution à 78% de la solution trouvée en trois heures de calcul par le programme linéaire.

Dans la prochaine section, nous allons développer une métaheuristique qui couple l'algorithme génétique et le recuit simulé en espérant profiter des points forts de chaque méthode .

7.4 Algorithme génétique couplé avec le recuit simulé

Le but est de profiter des avantages des deux métaheuristiques pour améliorer les résultats. Pour les instances industrielles, le recuit simulé donne rapidement des solutions intéressantes mais moins bonnes que les solutions données par l'algorithme génétique. L'algorithme génétique cherche au voisinage d'une population initiale générée aléatoirement pour l'améliorer, mais nécessite beaucoup d'effort de calcul.

7.4.1 Principe

Nous proposons de générer plusieurs solutions grâce à la rapidité du recuit simulé et de les considérer comme la population initiale de l'algorithme génétique. Nous espérons qu'avec la bonne qualité de cette population, l'algorithme génétique convergera plus rapidement. Nous notons ce couplage des deux méthodes GASA. Il est schématisé dans la figure 7.8.

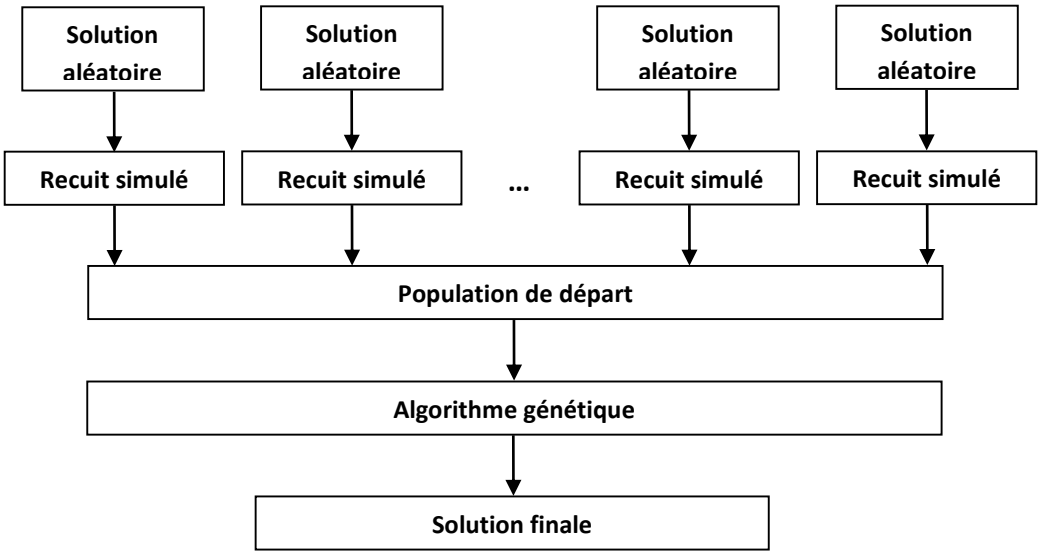


Figure 7.8 : Fonctionnement de GASA

Nous générons aléatoirement une solution initiale qui sera améliorée par le recuit simulé. La solution trouvée fera partie de la population de départ pour l'algorithme génétique. Nous appliquons le recuit simulé plusieurs fois (le nombre de chromosomes de la population de l'algorithme génétique). Cette population sera traitée par l'algorithme génétique pour donner une solution.

7.4.2 Résultats numériques

Dans cette section, nous testons GASA sur des instances mono-opérateur et multi-opérateurs afin de les comparer avec le programme linéaire et les deux premières métaheuristiques.

7.4.2.1 Cas mono-opérateur de type 1

Nous testons les instances académiques utilisées pour valider les deux premières métaheuristiques. En premier lieu nous testons 17 instances avec un seul opérateur chacune et 20 produits à séquencer (cf. tableaux 7.1 et 7.8). Le tableau 7.13 présente les résultats de ces tests. Nous faisons dix tests par instance.

Tableau 7.13 : Résultats des tests des instances académiques mono-opérateur 1

Instance	Programme linéaire		GASA					
	Solution optimale	Temps de calcul (s)	Objectif		Temps de calcul (s)		Gap	
			Pire	Meilleur	Plus long	Plus court	Pire	Meilleur
TH_MONO1_1	3,4	10,00	3,4	3,4	32,82	26,74	0%	0%
TH_MONO1_2	2,1	4,70	2,1	2,1	29,64	24,41	0%	0%
TH_MONO1_3	5,2	15,00	5,2	5,2	37,95	28,57	0%	0%
TH_MONO1_4	2,6	97,00	2,6	2,6	36,41	28,58	0%	0%
TH_MONO1_5	2,8	2,30	2,8	2,8	28,10	23,71	0%	0%
TH_MONO1_6	0,2	0,30	0,2	0,2	23,05	22,68	0%	0%
TH_MONO1_7	4,7	316,00	4,7	4,7	32,20	25,11	0%	0%
TH_MONO1_8	5,0	6,40	5,0	5,0	50,07	34,55	0%	0%
TH_MONO1_9	5,8	0,90	5,8	5,8	45,15	35,55	0%	0%
TH_MONO1_10	3,0	504,00	3,0	3,0	28,47	24,55	0%	0%
TH_MONO1_11	2,4	69,00	2,4	2,4	29,38	25,11	0%	0%
TH_MONO1_12	2,8	2,00	2,8	2,8	28,12	24,25	0%	0%
TH_MONO1_13	2,0	1,00	2,0	2,0	26,39	23,48	0%	0%
TH_MONO1_14	2,9	2,00	2,9	2,9	30,77	25,17	0%	0%
TH_MONO1_15	2,2	2,20	2,2	2,2	27,89	23,52	0%	0%
TH_MONO1_16	6,1	3,00	6,1	6,1	41,45	31,45	0%	0%
TH_MONO1_17	5,7	470,00	5,7	5,7	46,36	36,02	0%	0%

D'après le tableau 7.13, GASA trouve toujours la solution optimale pour les 17 instances. Il améliore donc la qualité des solutions obtenue par les deux précédentes métaheuristiques testées mais nécessite plus d'effort de calcul que les autres méthodes (algorithme génétique, recuit simulé et programmation linéaire). Le temps de calcul est d'environ 50 secondes pour le test le plus long.

En deuxième lieu, nous utilisons les instances académiques à 43 produits testées sur les deux premières métaheuristiques (tableaux 7.2 et 7.9). Nous rappelons que le programme linéaire n'arrive pas à les résoudre au bout de 30 minutes.

Le tableau 7.14 montre que GASA trouve en moins de temps les mêmes solutions que le programme linéaire. Cependant, le temps de calcul est plus important que celui du recuit simulé. Pour le cas mono-opérateur, GASA nécessite plus de temps de calcul que le recuit simulé pour trouver les mêmes solutions.

Tableau 7.14 : Résultats des tests des instances académiques mono-opérateur 2

Instance	Programme linéaire		GASA					
	Solution obtenue	Temps de calcul (s)	Objectif		Temps de calcul (s)		Gap	
			Pire	Meilleur	Plus long	Plus court	Pire	Meilleur
TH_MONO2_1	6,0	1800,00	6,0	6,0	45,19	35,49	0%	0%
TH_MONO2_2	8,4	1800,00	8,4	8,4	59,96	47,35	0%	0%
TH_MONO2_3	6,3	1800,00	6,3	6,3	51,80	29,17	0%	0%

Les résultats des tests mono-opérateur montrent que GASA donne des solutions de bonne qualité mais ces temps de calcul sont plus élevés que ceux du recuit simulé. Dans la prochaine section, nous testons GASA sur des instances multi-opérateurs pour vérifier si cette observation est confirmée pour ce type d'instances.

7.4.2.2 Cas multi-opérateurs de type 1

Nous testons les instances académiques générées aléatoirement pour valider l'algorithme génétique (tableau 7.3) et le recuit simulé (tableau 7.10). Les résultats sont exposés dans le tableau 7.15.

Tableau 7.15 : Résultats des tests des instances académiques multi-opérateur (type 1)

Instance	Programme linéaire		GASA					
	Solution optimale	Temps de calcul (s)	Objectif		Temps de calcul (s)		Gap	
			Pire	Meilleur	Plus long	Plus court	Pire	Meilleur
TH_MULTI_1	23,6	1373,00	23,6	23,6	54,09	41,72	0%	0%
TH_MULTI_2	16,0	136,00	16,2	16,0	49,04	39,79	1%	0%
TH_MULTI_3	5,5	311,00	5,5	5,5	43,80	33,20	0%	0%

GASA trouve presque toujours la solution optimale. La qualité des solutions trouvées est meilleure que celles obtenues par les deux premières heuristiques proposées. L'effort de calcul est moins important que pour le programme linéaire mais plus important que pour l'algorithme génétique et le recuit simulé.

7.4.2.3 Prise en compte des opérateurs de type 2 et 3

- **Instances académiques :**

Nous testons GASA sur les instances académiques utilisées pour tester l'algorithme génétique et le recuit simulé (cf. tableaux 7.4 et 7.11). Les résultats sont donnés dans le tableau 7.16. Ils montrent que GASA est plus performant que l'algorithme génétique seul et le recuit simulé seul mais il nécessite plus de temps de calcul. Le programme linéaire reste plus performant que les trois métaheuristiques proposées pour ces instances de petites tailles.

Tableau 7.16 : Résultats des tests académiques avec des opérateurs de type 2 et 3

Instance	Programme linéaire		GASA					
	Solution optimale	Temps de calcul (s)	Objectif		Temps de calcul (s)		Gap	
			Pire	Meilleur	Plus long	Plus court	Pire	Meilleur
1	6,02	0,60	6,28	6,02	68,97	66,37	4%	0%
2	10,20	0,30	10,20	10,20	64,52	63,33	0%	0%
3	19,46	1,76	19,94	19,46	138,95	131,81	2%	0%

- **Instances du cas d'étude**

Nous testons GASA sur les instances du cas d'étude (cf. tableaux 7.5 et 7.12). Les résultats sont présentés dans le tableau 7.17.

Tableau 7.17 : Résultats des tests des instances du cas d'étude

Instance	Programme linéaire		GASA					
	Solution optimale	Temps de calcul (s)	Objectif		Temps de calcul (s)		Gap	
			Pire	Meilleur	Plus long	Plus court	Pire	Meilleur
1	800,8	10800,00	822,9	782,7	1259,26	1189,75	3%	-2%
2	585,1	10800,00	593,0	572,1	1301,01	1208,93	1%	-2%
3	763,4	10800,00	774,2	763,2	1333,59	1187,14	1%	0%
4	914,5	10800,00	959,1	904,1	1226,15	1153,18	5%	-1%
5	1823,4	10800,00	1794,1	1711,7	1355,36	1252,93	-2%	-6%
6	804,4	10800,00	819,3	799,1	1397,61	1202,28	2%	-1%
7	814,8	10800,00	814,8	802,6	1345,02	1211,12	0%	-1%
8	1450,0	10800,00	1507,3	1428,9	1346,28	1225,53	4%	-1%
9	800,3	10800,0	766,0	757,3	1329,01	1193,41	-4%	-5%

Les résultats montrent que, dans les meilleurs cas, GASA améliore en à peu près 20 minutes les solutions trouvées par le programme linéaire en trois heures de calcul (de 1% à 6%). Dans les pires cas, GASA trouve des solutions proches de celles trouvées par le programme linéaire.

Les performances de GASA sont nettement meilleures que celles du recuit simulé et sont assez proches de celles de l'algorithme génétique. Dans les pires cas, GASA est un peu mieux que l'algorithme génétique mais le temps de calcul de GASA (environ 20 minutes) est plus important que celui de l'algorithme génétique. Dans les meilleurs cas, les performances de GASA sont similaires à celles de l'algorithme génétique. L'écart entre les solutions du meilleur cas et celles du pire cas est donc moins faible pour GASA. GASA demande un peu plus d'effort de calcul que l'algorithme génétique pour une petite amélioration la qualité des solutions. GASA améliore de 36% en moyenne la fonction objectif des séquences obtenues par rapport aux séquences de la procédure actuellement utilisée dans le cas d'étude.

7.5 Conclusion

Pour résoudre le problème de minimisation des retards cumulés d'une ligne de montage multi-modèles, nous avons développé trois métaheuristiques : l'algorithme génétique, le recuit simulé et une méthode qui couple les deux métaheuristiques (qu'on a appelé GASA)

L'algorithme génétique offre un bon compromis entre le temps de calcul et la qualité des solutions. Pour les instances du cas d'étude, il trouve des solutions proches du programme linéaire en un temps de calcul nettement moins important (entre 6 et 11 minutes). Il améliore de 35,7% en moyenne la fonction objectif des séquences obtenues par rapport aux séquences de la procédure actuellement utilisée dans le cas d'étude.

Le recuit simulé est performant pour les instances académiques (temps de calcul et qualité des solutions). Pour les instances du cas d'étude, il demande des temps de calcul (aux alentours de 80 secondes) moins important que l'algorithme génétique au détriment de la qualité des solutions. Dans le pire des cas, le recuit simulé atteint seulement 78% du résultat du programme linéaire. Dans ce travail, nous n'avons utilisé que l'inversion pour générer des voisinages pour le recuit simulé. Une des perspectives de cette partie est de tester d'autres types de voisinage.

GASA est un couplage entre l'algorithme génétique et le recuit simulé. Il est moins performant que le recuit simulé pour les instances académiques. Par contre, pour les instances du cas d'étude, GASA trouve en environ 20 minutes de calcul des solutions

proches ou meilleures que celle trouvées au bout de trois heures de calcul par le programme linéaire. Ses performances sont un peu meilleures que celle de l'algorithme génétique mais ne justifie pas l'effort de calcul. En effet, GASA améliore de 36% en moyenne la fonction objectif des séquences obtenues par rapport aux séquences de la procédure actuellement utilisée dans le cas d'étude (35,7% pour l'algorithme génétique) en des temps significativement plus élevés.

Chapitre 8 : Conclusion

Cette thèse a été consacrée au problème du séquencement des produits sur une ligne d'assemblage multi-modèles. Deux approches peuvent être utilisées pour résoudre ce problème : le *car sequencing* (respect des règles de séquencement) et le *mixed-model sequencing* (l'utilisation directe des temps opératoires). Cette dernière approche a reçu encore peu d'attention dans la littérature. Dans ce travail, nous avons choisi cette approche du *mixed-model sequencing* qui saisit mieux la diversité des temps opératoires, au prix d'une complexité plus élevée. L'analyse du cas industriel a montré que le critère d'optimisation approprié est la minimisation des retards cumulés.

L'étude de la littérature de ce problème a dévoilé une lacune dans les travaux précédents concernant les différents types d'opérateurs potentiels. En effet, l'essentiel des travaux modélisent un seul type d'opérateurs, opérateurs assignés à un seul pas de travail. La littérature a montré aussi que la majorité des articles proposent des heuristiques ou des métaheuristiques pour résoudre ces problèmes. L'originalité de notre travail est de tester des méthodes exactes pour des instances industrielles et de modéliser des opérateurs spécifiques au cas industriel (trois types d'opérateurs).

En premier lieu, nous avons donc testé des méthodes exactes pour résoudre le problème. Une modélisation en programmation linéaire a été proposée. L'analyse de complexité numérique a permis de déterminer les limites du programme linéaire en termes de nombre de produits, nombre d'opérateurs et de nombre de modèles. Pour les instances industrielles, nous n'obtenons pas de solution optimale au bout de trois heures de calcul. Cependant, les solutions trouvées améliorent nettement le critère d'optimisation par rapport à la procédure utilisée actuellement dans le cas d'étude.

Par ailleurs, nous avons proposé une modélisation en programmation dynamique. L'analyse de complexité numérique a permis de déterminer les limites du programme dynamique en termes de nombre de produits, nombre d'opérateurs et de nombre de modèles.

Des méthodes approchées ont été également développées afin de corriger la limite des méthodes exactes qui est le temps de calcul important. Une première heuristique basée sur la modélisation en programmation dynamique est proposée. Le principe de l'heuristique proposée est de couper certains des nœuds du graphe généré pour la programme dynamique afin de contrôler la taille du graphe à explorer. Cette heuristique est performante pour les instances académiques. Elle améliore le temps de calcul tout en

ayant des solutions proches de l'optimal. Cependant, l'heuristique est moins performante pour les tests du cas d'étude. Elle n'améliore pas les solutions trouvées par le programme linéaire.

Enfin, trois procédures basées sur des métaheuristiques ont été proposées pour résoudre le problème. La première est basée sur l'algorithme génétique, la deuxième sur le recuit simulé et la troisième est la combinaison des deux premières (GASA). L'algorithme génétique offre un bon compromis entre le temps de calcul et la qualité des solutions. Pour les instances du cas d'étude, l'algorithme génétique améliore en des temps de calcul raisonnables (6 à 11 minutes) de 35,7% en moyenne la fonction objectif des séquences obtenues par rapport aux séquences de la procédure actuellement utilisée dans le cas d'étude. Le recuit simulé demande pour les instances du cas d'étude des temps de calcul moins importants que l'algorithme génétique au détriment de la qualité des solutions. GASA trouve, pour les instances du cas d'étude, en environ 20 minutes de calcul, des solutions proches ou meilleures que celle trouvées par l'algorithme génétique mais ces performances ne justifie pas l'effort de calcul supplémentaire par rapport à l'algorithme génétique.

Nous avons donc proposé de nouveaux modèles d'optimisation (exactes et approchées) basés sur trois méthodes (programmation linéaire, programmation dynamique et des métaheuristiques) pour le problème de séquencement d'une ligne de montage multi-modèles. Ce travail a permis d'apporter une solution intéressante d'un point de vue industriel puisqu'il prend en compte les caractéristiques de la ligne de montage (opérateurs spécifiques) et améliore significativement la qualité du séquencement en un temps de calcul raisonnable.

Ce travail offre différentes perspectives. Tout d'abord, un prototype de l'outil de séquencement pourra être réalisé avec une interface Homme-Machine ergonomique et fiable. La définition des besoins de cet outil a été accomplie. L'outil devra permettre la saisie des données et des paramètres de l'optimisation, de visualiser les résultats selon des indicateurs pertinents (tels que l'opérateur le plus en retard, la position de la séquence qui engendre le plus de retard…). Lors de l'affichage des résultats, une interface permettra aussi de visualiser le respect des règles de séquencement afin de procéder à un passage hybride de la procédure actuelle à la procédure proposée pour faciliter le changement. Le développement du prototype et la formation des utilisateurs seront effectués sur la base de nos travaux.

A moyen et long termes, les perspectives de ce travail sont d'améliorer les méthodes développées dans ces travaux, d'étendre au cas où les postes ne sont pas indépendants, de

considérer plusieurs critères d'optimisation simultanément et d'intégrer l'équilibrage dynamique.

L'heuristique basée sur la programmation dynamique pourra être améliorée afin de donner des résultats plus intéressants. L'exploration du graphe en largeur constitue une piste à examiner. Les métaheuristiques donnent des bons résultats. Nous pouvons néanmoins faire varier les paramètres utilisés et explorer d'autres métaheuristiques.

Dans ces travaux, nous avons considéré que les opérateurs sont indépendants. Si un opérateur a un retard sur un véhicule celui-ci ne se répercute pas sur l'opérateur suivant. Cette hypothèse est issue de l'analyse du cas d'étude. Nous pourrons, dans des prochains travaux, développer de nouveaux modèles adaptés au cas où les opérateurs ne sont pas indépendants pour pouvoir les utiliser sur d'autres cas d'étude.

Le critère de minimisation des retards cumulé pourrait être associé à des critères de lissage de consommation des pièces afin de diminuer les stocks de sécurité et de faciliter la gestion de la logistique interne.

De plus, le séquencement pourrait intégrer l'équilibrage dynamique. En effet, l'équilibrage statique de la ligne est effectué à chaque changement de cadence (environ 3 mois) et est donc basé sur les données prévisionnelles de planification disponible à ce moment. Cependant, il y a un écart entre ces données et la réalité. Lors du séquencement, les produits à effectuer sont connus, on pourrait donc ajuster l'équilibrage statique en supprimant des tâches aux opérateurs les plus chargés et en les attribuant à des opérateurs moins chargés. Ceci permettra de lisser la charge des opérateurs tout en minimisant les retards.

Bibliographie

Aigbedo, H., et Monden, Y. (1997). A parametric procedure for multicriterion sequence scheduling for JustIn-Time mixed-model assembly lines. International Journal of Production Research *35*, 2543–2564.

Akgündüz, O.S., et Tunalı, S. (2009). An adaptive genetic algorithm approach for the mixed-model assembly line sequencing problem. International Journal of Production Research *48*, 5157–5179.

Albana, A.S., Aroui, K., Alpan, G., et Frein, Y. (2014). Mixed model assembly line sequencing to minimize delays using meta-heuristics. 1114–1127. In 44th International Conference on Computers and Industrial engineering (CIE 2014).

Aroui, K., Alpan, G., et Frein, Y. (2013). Minimisation des retards dans le séquencement des véhicules sur une ligne d'assemblage multi modèles. Journal Européen des Systèmes Automatisés *47*, 635–656.

Aroui, K., Alpan, G., et Frein, Y. (2014a). Minimizing work overload in mixed model assembly lines: A case study from truck industry. In 5th International conference on Information Systems, Logistics and Supply Chain.

Aroui, K., Alpan, G., Frein, Y. et Massonnet, G. (2014b). Exact and heuristic solutions based on dynamic programming for mixed model assembly line sequencing. In 44th International Conference on Computers and Industrial engineering (CIE 2014).

Bard, J.F., Dar-El, E., et Shtub, A. (1992). An analytic framework for sequencing mixed model assembly lines. International Journal of Production Research *30*, 35–48.

Bautista, J., et Cano, A. (2011). Solving mixed model sequencing problem in assembly lines with serial workstations with work overload minimisation and interruption rules. European Journal of Operational Research *210*, 495–513.

Bautista, J., et Cano, J. (2008). Minimizing work overload in mixed-model assembly lines. International Journal of Production Economics *112*, 177–191.

Bautista, J., Companys, R., et Corominas, A. (1996). Heuristics and exact algorithms for solving the Monden problem. European Journal of Operational Research *88*, 101–113.

Bautista, J., Cano, A., et Alfaro, R. (2012). Models for MMSP-W considering workstation dependencies: A case study of Nissan's Barcelona plant. European Journal of Operational Research *223*, 669–679.

Baybars, İ. (1986). A Survey of Exact Algorithms for the Simple Assembly Line Balancing Problem. Management Science *32*, 909–932.

Becker, C., et Scholl, A. (2006). A survey on problems and methods in generalized assembly line balancing. European Journal of Operational Research *168*, 694–715.

Bock, S., Rosenberg, O., et Brackel, T. van (2006). Controlling mixed-model assembly lines in real-time by using distributed systems. European Journal of Operational Research *168*, 880–904.

Bolat, A. (1997). Stochastic procedures for scheduling minimum job sets on mixed model assembly lines. Journal of the Operational Research Society *48*, 490–501.

Bolat, A., et Yano, C.A. (1992a). Scheduling algorithms to minimize utility work at a single station on a paced assembly line. Production Planning & Control *3*, 393–405.

Bolat, A., et Yano, C.A. (1992b). A surrogate objective for utility work in paced assembly lines. Production Planning & Control *3*, 406–412.

Bolat, A., Savsar, M., et Al-Fawzan, M.A. (1994). Algorithms for real-time scheduling of jobs on mixed model assembly lines. Computers & Operations Research *21*, 487–498.

Boysen, N., et Fliedner, M. (2006). Ein flexibler zweistufiger Graphen-Algorithmus zur Fließbandabstimmung mit praxisrelevanten Nebenbedingungen. Z. Betriebswirtsch *76*, 55–78.

Boysen, N., Fliedner, M., et Scholl, A. (2008). Assembly line balancing: Which model to use when? International Journal of Production Economics *111*, 509–528.

Boysen, N., Fliedner, M., et Scholl, A. (2009a). Sequencing mixed-model assembly lines: Survey, classification and model critique. European Journal of Operational Research *192*, 349–373.

Boysen, N., Fliedner, M., et Scholl, A. (2009b). The product rate variation problem and its relevance in real world mixed-model assembly lines. European Journal of Operational Research *197*, 818–824.

Boysen, N., Kiel, M., et Scholl, A. (2010). Sequencing mixed-model assembly lines to minimise the number of work overload situations. International Journal of Production Research *49*, 4735–4760.

Brailsford, S.C., Potts, C.N., et Smith, B.M. (1999). Constraint satisfaction problems: Algorithms and applications. European Journal of Operational Research *119*, 557–581.

Cano-Belmán, J., Ríos-Mercado, R.Z., et Bautista, J. (2010). A scatter search based hyper-heuristic for sequencing a mixed-model assembly line. J Heuristics *16*, 749–770.

Celano, G., Costa, A., Fichera, S., et Perrone, G. (2004). Human factor policy testing in the sequencing of manual mixed model assembly lines. Computers & Operations Research *31*, 39–59.

Chutima, P., Nimmano, W., et Yiangkamolsing, C. (2003). Application of fuzzy genetic algorithm for sequencing in mixed-model assembly line with processing time. International Journal of Industrial Engineering 10, 325–331.

Comby, G. (1996). « Aide au séquencement des produits sur une ligne de fabrication multi modèles ». Thèse de doctorat, Institut National des Sciences Appliquées de Lyon, soutenue le 19/12/1996.

Cortez, P.M.C., et Costa, A.M. (2014). Sequencing mixed-model assembly lines operating with a heterogeneous workforce. International Journal of Production Research 2014, 1–14.

Decker, M. (1993). Capacity smoothing and sequencing for mixed-model lines. International Journal of Production Economics *30–31*, 31–42.

Domschke, W. (1996). Antizipative Leistungsabstimmung bei moderner Variantenfließfertigung. Zeitschrift für Betriebswirtschaft, 66 1465–1490.

Dong, J., Zhang, L., Xiao, T., et Mao, H. (2014). Balancing and sequencing of stochastic mixed-model assembly U-lines to minimise the expectation of work overload time. International Journal of Production Research *52*, 7529–7548.

Drexl, A., et Kimms, A. (2001). Sequencing JIT Mixed-Model Assembly Lines Under Station-Load and Part-Usage Constraints. Management Science *47*, 480–491.

Dar-El, E.M. (1978). Mixed-model assembly line sequencing problems. Omega *6*, 313–323.

Dar-El, E., et Cucuy, S. (1977). Optimal mixed-model sequencing for balanced assembly lines. Omega *5*, 333–342.

Dar-El, E.M., et Cother, R.F. (1975). Assembly line sequencing for model mix. International Journal of Production Research *13*, 463–477.

Dar-El, E.M., et Nadivi, A. (1981). A mixed-model sequencing application. International Journal of Production Research *19*, 69–84.

Fattahi, P., et Salehi, M. (2009). Sequencing the mixed-model assembly line to minimize the total utility and idle costs with variable launching interval. Int J Adv Manuf Technol *45*, 987–998.

Gagné, C., Gravel, M., et Price, W.L. (2006). Solving real car sequencing problems with ant colony optimization. European Journal of Operational Research *174*, 1427–1448.

Giard, V. (2003). Gestion de la production et des flux (Economica).

Giard, V., et Jeunet, J. (2004). Modélisation du problème général d'ordonnancement de véhicules sur une ligne de production et d'assemblage. Journal Européen des Systèmes Automatisés *40*, 4 63-496.

Goldberg, D.E., et Holland, J.H. (1988). Genetic Algorithms and. Machine Learning *3*, 95–99.

Goldschmidt, O., Bard, J.F., et Takvorian, A. (1997). Complexity Results for Mixed-Model Assembly Lines with Approximation Algorithms for the Single Station Case. International Journal of Flexible Manufacturing Systems *9*, 251–272.

Golle, U., Boysen, N., et Rothlauf, F. (2010). Analysis and design of sequencing rules for car sequencing. European Journal of Operational Research *206*, 579–585.

Golle, U., Rothlauf, F., et Boysen, N. (2014). Car sequencing versus mixed-model sequencing: A computational study. European Journal of Operational Research *237*, 50–61.

Gottlieb, J., Puchta, M., et Solnon, C. (2003). A Study of Greedy, Local Search, and Ant Colony Optimization Approaches for Car Sequencing Problems. In Applications of Evolutionary Computing, S. Cagnoni, C.G. Johnson, J.J.R. Cardalda, E. Marchiori, D.W. Corne, J.-A. Meyer, J. Gottlieb, M. Middendorf, A. Guillot, G.R. Raidl, et al., eds. (Springer Berlin Heidelberg), pp. 246–257.

Gravel, M., Gagné, C., et Price, W.L. (2005). Review and comparison of three methods for the solution of the car sequencing problem. J Oper Res Soc *56*, 1287–1295.

Hamzadayi, A., et Yildiz, G. (2012). A genetic algorithm based approach for simultaneously balancing and sequencing of mixed-model U-lines with parallel workstations and zoning constraints. Computers & Industrial Engineering *62*, 206–215.

Hamzadayi, A., et Yildiz, G. (2013). A simulated annealing algorithm based approach for balancing and sequencing of mixed-model U-lines. Computers & Industrial Engineering *66*, 1070–1084.

Holland, J.H. (1975). Adaptation in natural and artificial systems: An introductory analysis with applications to biology, control, and artificial intelligence (Oxford, England: U Michigan Press).

Hwang, R., et Katayama, H. (2009). Integrated procedure of balancing and sequencing for mixed-model assembly lines: a multi-objective evolutionary approach. International Journal of Production Research *48*, 6417–6441.

Javadi, B., Rahimi-Vahed, A., Rabbani, M., et Dangchi, M. (2008). Solving a multi-objective mixed-model assembly line sequencing problem by a fuzzy goal programming approach. Int J Adv Manuf Technol *39*, 975–982.

Kara, Y. (2008). Line balancing and model sequencing to reduce work overload in mixed-model U-line production environments. Engineering Optimization *40*, 669–684.

Khadka, S.R., et Werner, F. (2013). Upper and Lower Bounds for the Total Product Rate Variation Problem.

Kim, S., et Jeong, B. (2007). Product sequencing problem in Mixed-Model Assembly Line to minimize unfinished works. Computers & Industrial Engineering *53*, 206–214.

Kim, Y.K., Hyun, C.J., et Kim, Y. (1996). Sequencing in mixed model assembly lines: A genetic algorithm approach. Computers & Operations Research *23*, 1131–1145.

Kirkpatrick, S., Gelatt, C.D., et Vecchi, M.P. (1983). Optimization by Simulated Annealing. Science *220*, 671–680.

Kotani, S., Ito, T., et Ohno, K. (2004). Sequencing problem for a mixed-model assembly line in the Toyota production system. International Journal of Production Research *42*, 4955–4974.

Kubiak, W. (1993). Minimizing variation of production rates in just-in-time systems: A survey. European Journal of Operational Research *66*, 259–271.

Lesert, A. (2006). « Sur l'évaluation de la flexibilité de l'atelier montage d'une usine terminale automobile ». Thèse de doctorat, Institut National Polytechnique de Grenoble, soutenue le 18/12/2006.

Lesert, A., Alpan, G., Frein, Y., et Noiré, S. (2011). Definition of spacing constraints for the car sequencing problem. International Journal of Production Research *49*, 963–994.

McMullen, P.R., et Frazier, G.V. (1997). A heuristic for solving mixed-model line balancing problems with stochastic task durations and parallel stations. International Journal of Production Economics *51*, 177–190.

Miltenburg, J. (1989). Level Schedules for Mixed-Model Assembly Lines in Just-In-Time Production Systems. Management Science *35*, 192–207.

Monden, Y. (1983). Toyota Production System (Industrial Engineering and Management Press, Institute of Industrial Engineers, Atlanta, GA).

Monden, Y. (2011). Toyota Production System: An Integrated Approach to Just-In-Time, 4th Edition (CRC Press).

Mosadegh, H., Zandieh, M., et Ghomi, S.M.T.F. (2012). Simultaneous solving of balancing and sequencing problems with station-dependent assembly times for mixed-model assembly lines. Applied Soft Computing *12*, 1359–1370.

Nguyen, A., et Cung, V.-D. (2005). Le problème du Car Sequencing RENAULT et le Challenge ROADEF'2005. (Université d'Artois), pp. 3–10.

Okamura, K., et Yamashina, H. (1979). A heuristic algorithm for the assembly line model-mix sequencing problem to minimize the risk of stopping the conveyor. International Journal of Production Research *17*, 233–247.

Özcan, U., Kellegöz, T., et Toklu, B. (2010). A genetic algorithm for the stochastic mixed-model U-line balancing and sequencing problem. International Journal of Production Research *49*, 1605–1626.

Özcan, U., Kellegöz, T., et Toklu, B. (2011). A genetic algorithm for the stochastic mixed-model U-line balancing and sequencing problem. International Journal of Production Research *49*, 1605–1626.

Parrello, B. (1988). Car Wars: (Almost) Birth of an Expert System. AI Expert *3*, 60–64.

Pastor, R., Andrés, C., Duran, A., et Pérez, M. (2002). Tabu Search Algorithms for an Industrial Multi-Product and Multi-Objective Assembly Line Balancing Problem, with Reduction of the Task Dispersion. The Journal of the Operational Research Society *53*, 1317–1323.

Rabbani, M., Rahimi-Vahed, A., et Torabi, S.A. (2008). Real options approach for a mixed-model assembly line sequencing problem. Int J Adv Manuf Technol *37*, 1209–1219.

Rahimi-Vahed, A., et Mirzaei, A.H. (2007). A hybrid multi-objective shuffled frog-leaping algorithm for a mixed-model assembly line sequencing problem. Computers & Industrial Engineering *53*, 642–666.

Rahimi-Vahed, A.R., Rabbani, M., Tavakkoli-Moghaddam, R., Torabi, S.A., et Jolai, F. (2007). A multi-objective scatter search for a mixed-model assembly line sequencing problem. Advanced Engineering Informatics *21*, 85–99.

Roberts, S.D., et Villa, C.D. (1970). On a Multiproduct Assembly Line-Balancing Problem. AIIE Transactions *2*, 361–364.

Saif, U., Guan, Z., Liu, W., Wang, B., et Zhang, C. (2014). Multi-objective artificial bee colony algorithm for simultaneous sequencing and balancing of mixed model assembly line. Int J Adv Manuf Technol *75*, 1809–1827.

Santosa, B., et Willy, P. (2011). Metoda Metaheuristik Konsep dan Implementasi, Surabaya, Guna Widya.

Sarker, B.R., et Pan, H. (1998). Designing a mixed-model assembly line to minimize the costs of idle and utility times. Computers & Industrial Engineering *34*, 609–628.

Sarker, B.R., et Pan, H. (2001). Designing a Mixed-Model, Open-Station Assembly Line Using Mixed-Integer Programming. The Journal of the Operational Research Society *52*, 545–558.

Scholl, A. (1999). Balancing and sequencing assembly lines, 2nd ed. Physica, Heidelberg.

Scholl, A., Klein, R., et Domschke, W. (1998). Pattern Based Vocabulary Building for Effectively Sequencing Mixed-Model Assembly Lines. Journal of Heuristics *4*, 359–381.

Solnon, C., Cung, V.D., Nguyen, A., et Artigues, C. (2008). The car sequencing problem: Overview of state-of-the-art methods and industrial case-study of the ROADEF'2005 challenge problem. European Journal of Operational Research *191*, 912–927.

Sumichrast, R.T., Oxenrider, K.A., et Clayton, E.R. (2000). An Evolutionary Algorithm for Sequencing Production on a Paced Assembly Line. Decision Sciences *31*, 149–172.

Tavakkoli-Moghaddam, R., et Rahimi-Vahed, A.R. (2006). Multi-criteria sequencing problem for a mixed-model assembly line in a JIT production system. Applied Mathematics and Computation *181*, 1471–1481.

Thomopoulos, N.T. (1970). Mixed Model Line Balancing with Smoothed Station Assignments. Management Science *16*, 593–603.

Tsai, L.-H. (1995). Mixed-Model Sequencing to Minimize Utility Work and the Risk of Conveyor Stoppage. Management Science *41*, 485–495.

Vilarinho, P.M., et Simaria, A.S. (2002). A two-stage heuristic method for balancing mixed-model assembly lines with parallel workstations. International Journal of Production Research *40*, 1405–1420.

Villeminot, A. (2004). « Modélisation et simulation de la logistique d'approvisionnement dans l'industrie automobile : application pour un grand constructeur » Thèse de doctorat, Université Henri Poincaré, Nancy-I, soutenue le 10/12/2004.

Wee, T.S., et Magazine, M.J. (1982). Assembly line balancing as generalized bin packing. Operations Research Letters *1*, 56–58.

Wester, L., et Kilbridge, M. (1964). The assembly line model-mix sequencing problem. Third international conference on Operation Research, Oslo 1963, pp. 247–260.

Xiaobo, Z., et Ohno, K. (1994). A sequencing problem for a mixed-model assembly line in a JIT production system. Computers & Industrial Engineering *27*, 71–74.

Xiaobo, Z., et Ohno, K. (1997). Algorithms for sequencing mixed models on an assembly line in a JIT production system. Computers & Industrial Engineering *32*, 47–56.

Xiaobo, Z., et Ohno, K. (2000). Properties of a sequencing problem for a mixed model assembly line with conveyor stoppages. European Journal of Operational Research *124*, 560–570.

Yang, X.-S. (2010). Nature-inspired Metaheuristic Algorithms (Luniver Press).

Yano, C.A., et Rachamadugu, R. (1991). Sequencing to Minimize Work Overload in Assembly Lines with Product Options. Management Science *37*, 572–586.

Yoo, J.K., Shimizu, Y., et Hino, R. (2005). A Sequencing Problem for Mixed-Model Assembly Line with the Aid of Relief-Man. JSME International Journal Series C Mechanical Systems, Machine Elements and Manufacturing *48*, 15–20.

Annexes

Annexe A : Propriété de la solution optimale du cas mono-opérateur

Annexe B : Résultats des tests sur les instances issues de la littérature

Annexe C : Procédure de génération des données pour l'analyse numérique de la complexité

Annexe D : La performance de l'heuristique basée sur la programmation dynamique pour les tests du cas d'étude

Annexe A : Propriété de la solution optimale du cas mono-opérateur

Nous allons démontrer la deuxième propriété de la solution optimale, énoncée dans le chapitre 5 (section 5.3.3) qui est : « Dans une sous-séquence de produits ayant des temps opératoires inférieurs au temps de cycle, ces produits sont classés dans l'ordre croissant de leurs temps opératoires ».

Supposons qu'on ait deux modèles *a* et *b* à séquencer aux positions *j* et *j+1* tels que $t_a<t_b<\gamma$. Notons w_{j-1} le retard cumulé jusqu'à la position *j-1*.

Les repos des modèles a et b sont:

$d_a=t_a-\gamma<0$ et $d_b=t_b-\gamma<0$

1. On classe les produits dans l'ordre croissant des temps opératoires c'est-à-dire le modèle *a* à la position *j* et le modèle *b* à la position *j+1*.

Calculons le retard cumulé à chaque position que nous appelons $w_j(crois)$ puisque les produits sont classés dans l'ordre croissant:

$w_j(crois) = \max(0 , c_j)$ avec $c_j=w_{j-1} + d_a$

$w_{j+1}(crois) = \max(0 , c_{j+1}) = \max(0, w_j(crois) + d_b)$

$= \max(0, \max(0, w_{j-1} + d_a)+d_b)$

2. On classe les produits dans l'ordre décroissant des temps opératoires c'est-à-dire le modèle *b* à la position *j* et le modèle *a* à la position *j+1*.

Calculons le retard cumulé à chaque position que nous appelons $w_j(décr)$ puisque les produits sont classés dans l'ordre décroissant:

$w_j(décr) = \max(0 , c_j)$ avec $c_j = w_{j-1} + d_b$

$w_{j+1}(décr) = \max(0 , c_{j+1}) = \max(0 , w_j(décr)+d_a)$

$= \max(0, \max(0, w_{j-1} + d_b)+d_a)$

Nous allons traiter tous les cas possibles et vérifier quel classement minimise la fonction objectif :

- Si $w_{j-1} + d_a > 0$:

Alors $w_j(crois) = w_{j-1} + d_a$ et $w_{j+1}(crois) = \max(0, w_{j-1} + d_a + d_b)$

- Si $w_{j-1} + d_a + d_b > 0$

$f(croissant) = w_j(crois) + w_{j+1}(crois) = 2.w_{j-1} + 2.d_a + d_b$

- Si $w_{j-1} + d_b > 0$

alors $w_j(décr) = w_{j-1} + d_b$

et $w_{j+1}(décr) = max(0, w_{j-1} + d_b + d_a) = w_{j-1} + d_b + d_a$

$f(décroissant) = w_j(décr) + w_{j+1}(décr) = 2.w_{j-1} + d_a + 2.d_b$

$d_b > d_a$ alors $f(décroissant) > f(croissant)$

- Si $w_{j-1} + d_b \leq 0$

Impossible car $w_{j-1} + d_b + d_a > 0$ et $d_a \leq 0$

- Si $w_{j-1} + d_a + d_b \leq 0$

alors $w_{j+1}(crois)=0$ et $f(croissant) = w_{j-1} + d_a$

- Si $w_{j-1} + d_b > 0$

alors $w_j(décr) = w_{j-1} + d_b$ et $w_{j+1}(décr)=0$ et $f(décroissant)=w_{j-1} + d_b$

$d_b > d_a$ alors $f(décroissant) > f(croissant)$

- Si $w_{j-1} + d_b \leq 0$

Impossible car $w_{j-1} + d_a > 0$ et $d_b > d_a$

- Si $w_{j-1} + d_a \leq 0$:

alors $w_j(crois)=0$ et $w_{j+1}(crois) = \max(0 , d_b)=0$

$f(décroissant) = f(croissant) = 0$

Dans tous les cas $\boldsymbol{f}$**(décroissant)** $\geq$ $\boldsymbol{f}$**(croissant)** alors le séquencement dans l'ordre croissant des temps opératoires minimise la fonction objectif.

Annexe B : Résultats des tests sur les instances issues de la littérature

Cette étude expérimentale est basée sur 225 instances issues de l'étude de Bautista et Cano (2008). Ces instances ont été testées en utilisant la programmation linéaire (chapitre 5, section 5.4.1.1) et l'algorithme génétique (chapitre 7, section 7.2.7.2). Les temps opératoires et les programmes de production sont donnés dans les tableaux 5.9 et 5.10. Nous rappelons que la première structure présente des temps opératoires proches du temps de cycle.

Tableau B.1 : Structure 1

Programme	Programme linéaire		Algorithme génétique					
	Solution optimale	Temps de calcul (s)	Objectif		Temps de calcul (s)		Qualité	
			Pire	Meilleur	Plus long	Plus court	Pire	Meilleure
1	199,0	0,06	200,0	199,0	7,76	4,49	1%	0%
2	566,0	0,06	567,0	566,0	5,16	3,36	0%	0%
3	402,0	0,05	412,0	402,0	4,96	3,60	2%	0%
4	850,0	0,03	850,0	850,0	5,35	3,39	0%	0%
5	165,0	0,14	165,0	165,0	25,36	15,81	0%	0%
6	202,0	0,06	205,0	202,0	18,40	9,88	1%	0%
7	0,0	0,11	3,0	0,0	34,84	22,88	∞	∞
8	140,0	0,16	141,0	140,0	34,89	24,60	1%	0%
9	368,0	0,08	369,0	368,0	22,24	16,06	0%	0%
10	322,0	0,13	345,0	322,0	26,21	16,42	7%	0%
11	16,0	2,56	24,0	16,0	36,14	25,08	50%	0%
12	20,0	0,89	24,0	20,0	24,84	18,89	20%	0%
13	6,0	4,74	9,0	6,0	32,47	25,07	50%	0%
14	16,0	3,10	28,0	16,0	33,68	24,95	75%	0%
15	32,0	3,60	42,0	32,0	31,47	22,89	31%	0%
16	47,0	1,17	64,0	48,0	28,63	22,60	36%	2%
17	9,0	18,49	9,0	9,0	32,25	25,67	0%	0%
18	137,0	0,22	137,0	137,0	24,00	12,12	0%	0%
19	28,0	1,51	35,0	28,0	35,26	22,96	25%	0%
20	30,0	2,72	39,0	31,0	28,64	18,15	30%	3%
21	125,0	0,75	140,0	125,0	23,70	17,82	12%	0%
22	218,0	0,16	229,0	218,0	19,88	13,99	5%	0%
23	149,0	0,20	162,0	149,0	29,44	21,51	9%	0%
24	251,0	0,16	254,0	251,0	16,49	10,80	1%	0%
25	79,0	0,84	97,0	79,0	33,61	20,43	23%	0%
26	198,0	0,14	202,0	198,0	22,86	20,01	2%	0%
27	105,0	0,67	110,0	105,0	28,90	16,03	5%	0%
28	135,0	0,64	140,0	137,0	28,83	18,91	4%	1%
29	120,0	0,51	133,0	120,0	30,94	18,46	11%	0%
30	149,0	0,61	156,0	149,0	26,91	19,48	5%	0%
31	133,0	0,11	143,0	133,0	20,47	13,14	8%	0%
32	138,0	0,75	149,0	138,0	26,44	17,04	8%	0%
33	133,0	0,09	142,0	133,0	31,80	24,25	7%	0%
34	29,0	1,11	32,0	29,0	29,95	20,20	10%	0%
35	74,0	0,11	88,0	74,0	21,87	15,38	19%	0%
36	30,0	0,76	33,0	30,0	31,82	24,21	10%	0%
37	140,0	0,08	149,0	140,0	21,03	14,97	6%	0%
38	66,0	0,75	70,0	66,0	21,14	14,85	6%	0%
39	143,0	0,13	152,0	143,0	34,16	24,72	6%	0%
40	7,0	1,25	9,0	7,0	30,12	18,42	29%	0%
41	60,0	0,12	64,0	60,0	20,29	11,78	7%	0%
42	1,0	1,12	1,0	1,0	36,39	26,97	0%	0%
43	147,0	0,08	147,0	147,0	14,09	10,43	0%	0%
44	24,0	1,23	28,0	24,0	30,05	22,17	17%	0%
45	147,0	0,11	149,0	147,0	22,46	15,19	1%	0%

La deuxième structure présente des temps opératoires distants du temps de cycle.

Tableau B.2 : Structure 2

Programme	Programme linéaire		Algorithme génétique					
	Solution optimale	Temps de calcul (s)	Solution optimale		Temps de calcul (s)		Solution optimale	
			Pire	Meilleur	Plus long	Plus court	Pire	Meilleure
1	1331,0	0,06	1336,0	1331,0	4,32	3,19	0%	0%
2	1662,0	0,03	1679,0	1662,0	4,80	3,09	1%	0%
3	1646,0	0,06	1646,0	1646,0	5,28	3,39	0%	0%
4	1838,0	0,03	1863,0	1838,0	4,70	3,59	1%	0%
5	1021,0	0,08	1021,0	1021,0	26,28	20,60	0%	0%
6	404,0	0,12	421,0	404,0	28,98	16,59	4%	0%
7	634,0	0,09	634,0	634,0	33,00	24,43	0%	0%
8	1150,0	0,08	1150,0	1150,0	24,79	17,52	0%	0%
9	1204,0	0,09	1204,0	1204,0	31,94	21,22	0%	0%
10	1240,0	0,09	1240,0	1240,0	31,75	27,57	0%	0%
11	366,0	0,76	379,0	366,0	23,07	15,44	4%	0%
12	266,0	1,72	280,0	266,0	31,27	23,53	5%	0%
13	273,0	1,26	291,0	273,0	31,68	24,50	7%	0%
14	475,0	0,84	489,0	476,0	30,48	16,97	3%	0%
15	432,0	1,20	474,0	432,0	31,65	22,13	10%	0%
16	453,0	0,62	509,0	453,0	27,44	22,34	12%	0%
17	321,0	2,29	347,0	321,0	26,24	22,74	8%	0%
18	525,0	0,09	525,0	525,0	18,50	13,70	0%	0%
19	607,0	0,72	614,0	607,0	27,77	17,20	1%	0%
20	257,0	1,70	269,0	257,0	34,38	24,47	5%	0%
21	864,0	0,14	864,0	864,0	31,48	21,26	0%	0%
22	943,0	0,11	943,0	943,0	23,48	17,73	0%	0%
23	919,0	0,12	919,0	919,0	27,66	19,84	0%	0%
24	972,0	0,08	972,0	972,0	32,23	23,95	0%	0%
25	916,0	0,09	970,0	916,0	35,04	23,18	6%	0%
26	987,0	0,11	987,0	987,0	22,65	18,11	0%	0%
27	910,0	0,41	910,0	910,0	26,54	17,34	0%	0%
28	593,0	0,14	639,0	593,0	31,51	17,04	8%	0%
29	597,0	0,51	615,0	597,0	26,49	20,99	3%	0%
30	723,0	0,17	755,0	723,0	31,10	24,12	4%	0%
31	683,0	0,06	683,0	683,0	28,09	20,95	0%	0%
32	741,0	0,09	754,0	741,0	22,95	15,84	2%	0%
33	675,0	0,11	675,0	675,0	24,48	17,81	0%	0%
34	320,0	0,81	342,0	320,0	23,73	15,36	7%	0%
35	286,0	1,00	286,0	286,0	23,47	18,58	0%	0%
36	559,0	0,62	581,0	559,0	30,84	22,58	4%	0%
37	442,0	0,11	486,0	442,0	29,19	24,26	10%	0%
38	784,0	0,13	788,0	784,0	16,93	13,44	1%	0%
39	716,0	0,14	726,0	716,0	15,07	10,72	1%	0%
40	247,0	1,59	262,0	247,0	30,72	23,00	6%	0%
41	240,0	1,92	256,0	240,0	33,93	22,68	7%	0%
42	607,0	0,59	619,0	607,0	26,58	17,53	2%	0%
43	450,0	0,11	491,0	450,0	24,92	17,54	9%	0%
44	743,0	0,19	759,0	743,0	30,16	21,96	2%	0%
45	618,0	0,11	618,0	618,0	28,50	19,70	0%	0%

Nous rappelons que la troisième structure a un déséquilibre entre les deux premiers et les deux derniers opérateurs.

Tableau B.3 : Structure 3

Programme	Programme linéaire		Algorithme génétique					
	Solution optimale	Temps de calcul (s)	Solution optimale		Temps de calcul (s)		Solution optimale	
			Pire	Meilleur	Plus long	Plus court	Pire	Meilleure
1	2138,0	0,02	2175,0	2138,0	5,59	3,31	2%	0%
2	1828,0	0,03	1874,0	1828,0	4,83	3,63	3%	0%
3	2472,0	0,03	2472,0	2472,0	4,64	3,72	0%	0%
4	1700,0	0,01	1709,0	1700,0	5,55	3,78	1%	0%
5	1912,0	0,08	1912,0	1912,0	7,71	6,35	0%	0%
6	152,0	0,75	154,0	152,0	31,04	25,01	1%	0%
7	320,0	0,16	323,0	320,0	26,44	18,34	1%	0%
8	827,0	0,13	838,0	836,0	27,09	20,20	1%	1%
9	987,0	0,06	994,0	987,0	26,65	21,01	1%	0%
10	1943,0	0,03	1943,0	1943,0	8,36	5,60	0%	0%
11	370,0	0,70	370,0	370,0	18,81	15,80	0%	0%
12	185,0	0,98	255,0	185,0	27,01	22,65	38%	0%
13	197,0	0,20	197,0	197,0	29,39	22,64	0%	0%
14	412,0	0,87	426,0	412,0	30,55	25,06	3%	0%
15	526,0	0,16	551,0	526,0	24,16	14,66	5%	0%
16	508,0	0,16	536,0	508,0	24,44	14,20	6%	0%
17	282,0	1,11	303,0	282,0	34,12	19,18	7%	0%
18	302,0	0,25	302,0	302,0	20,90	13,99	0%	0%
19	560,0	0,09	560,0	560,0	18,65	14,23	0%	0%
20	270,0	0,17	299,0	270,0	26,76	19,40	11%	0%
21	986,0	0,42	1002,0	986,0	20,13	14,45	2%	0%
22	1316,0	0,08	1383,0	1316,0	13,19	8,95	5%	0%
23	1397,0	0,06	1446,0	1397,0	12,29	10,07	4%	0%
24	971,0	0,09	977,0	971,0	14,41	9,42	1%	0%
25	1031,0	0,09	1045,0	1031,0	19,36	15,77	1%	0%
26	872,0	0,14	895,0	872,0	27,45	20,09	3%	0%
27	814,0	0,14	852,0	814,0	26,19	18,69	5%	0%
28	831,0	0,13	864,0	831,0	14,72	11,69	4%	0%
29	914,0	0,11	932,0	914,0	16,92	11,50	2%	0%
30	544,0	0,09	544,0	544,0	25,97	21,34	0%	0%
31	433,0	0,14	447,0	433,0	28,47	22,72	3%	0%
32	625,0	0,09	654,0	625,0	14,35	11,33	5%	0%
33	448,0	0,20	468,0	448,0	23,75	16,39	4%	0%
34	245,0	0,87	275,0	245,0	22,05	16,66	12%	0%
35	315,0	0,17	326,0	322,0	23,51	16,40	3%	2%
36	327,0	0,16	351,0	327,0	24,21	16,36	7%	0%
37	211,0	0,16	211,0	211,0	35,80	27,45	0%	0%
38	1095,0	0,08	1095,0	1095,0	12,42	9,52	0%	0%
39	834,0	0,06	834,0	834,0	15,95	11,78	0%	0%
40	160,0	0,62	160,0	160,0	32,93	24,38	0%	0%
41	136,0	0,78	177,0	136,0	30,14	23,19	30%	0%
42	580,0	0,12	580,0	580,0	20,03	15,55	0%	0%
43	314,0	0,16	314,0	314,0	30,98	21,41	0%	0%
44	1123,0	0,14	1123,0	1123,0	14,52	12,24	0%	0%
45	862,0	0,09	862,0	862,0	16,28	13,28	0%	0%

La quatrième structure est caractérisée par le fait que deux modèles peuvent provoquer des retards à tous les opérateurs.

Tableau B.4 : Structure 4

Programme	Programme linéaire		Algorithme génétique					
	Solution optimale	Temps de calcul (s)	Solution optimale		Temps de calcul (s)		Solution optimale	
			Pire	Meilleur	Plus long	Plus court	Pire	Meilleure
1	4280,0	1,05	4307,0	4280,0	5,04	3,33	1%	0%
2	3775,0	0,03	3799,0	3775,0	4,88	3,63	1%	0%
3	0,0	0,02	0,0	0,0	2,73	1,86	∞	∞
4	0,0	0,44	0,0	0,0	2,43	1,98	∞	∞
5	3937,0	0,08	3961,0	3937,0	7,93	6,41	1%	0%
6	46,0	0,86	46,0	46,0	29,41	22,52	0%	0%
7	284,0	0,52	298,0	284,0	19,87	15,33	5%	0%
8	42,0	1,01	46,0	42,0	31,19	21,72	10%	0%
9	215,0	0,42	225,0	215,0	24,42	18,96	5%	0%
10	0,0	0,00	0,0	0,0	3,44	1,94	∞	∞
11	543,0	0,22	573,0	543,0	24,12	13,95	6%	0%
12	43,0	8,44	48,0	44,0	28,63	22,13	12%	2%
13	95,0	2,75	111,0	95,0	27,31	17,82	17%	0%
14	38,0	11,37	39,0	38,0	28,75	23,44	3%	0%
15	0,0	0,40	1,0	0,0	19,68	13,54	∞	∞
16	1,0	8,03	1,0	1,0	18,50	12,50	0%	0%
17	49,0	11,33	60,0	49,0	32,26	24,10	22%	0%
18	459,0	0,69	459,0	459,0	26,60	21,84	0%	0%
19	842,0	0,42	878,0	842,0	21,42	15,86	4%	0%
20	0,0	0,08	1,0	0,0	19,47	13,99	∞	∞
21	0,0	0,19	1,0	0,0	21,05	14,50	∞	∞
22	0,0	0,02	0,0	0,0	5,93	3,41	∞	∞
23	0,0	0,02	0,0	0,0	6,06	3,26	∞	∞
24	3,0	4,54	5,0	3,0	20,00	14,78	67%	0%
25	0,0	0,00	0,0	0,0	26,62	14,93	∞	∞
26	70,0	1,67	88,0	70,0	30,87	21,50	26%	0%
27	44,0	3,73	47,0	46,0	30,97	22,19	7%	5%
28	0,0	0,05	0,0	0,0	5,13	3,35	∞	∞
29	0,0	0,02	0,0	0,0	5,23	3,55	∞	∞
30	229,0	0,66	234,0	229,0	26,67	18,91	2%	0%
31	33,0	2,81	33,0	33,0	32,11	23,30	0%	0%
32	793,0	0,08	793,0	793,0	25,34	20,38	0%	0%
33	435,0	0,14	435,0	435,0	25,30	18,40	0%	0%
34	3,0	2,43	4,0	3,0	19,61	14,32	33%	0%
35	0,0	0,06	0,0	0,0	22,31	14,83	∞	∞
36	255,0	0,86	265,0	255,0	24,34	17,42	4%	0%
37	33,0	2,89	33,0	33,0	29,84	22,28	0%	0%
38	1955,0	0,09	1981,0	1955,0	14,19	9,01	1%	0%
39	1687,0	0,13	1733,0	1687,0	13,63	10,88	3%	0%
40	121,0	1,31	135,0	121,0	25,55	19,10	12%	0%
41	66,0	2,00	67,0	66,0	33,08	22,99	2%	0%
42	901,0	0,14	939,0	901,0	16,20	11,08	4%	0%
43	499,0	0,17	509,0	499,0	18,94	15,21	2%	0%
44	2022,0	0,08	2052,0	2022,0	11,69	9,42	1%	0%
45	1743,0	0,13	1789,0	1743,0	14,47	10,73	3%	0%

Dans la cinquième structure, chaque opérateur a des temps opératoires importants pour un modèle donné.

Tableau B.5 : Structure 5

Programme	Programme linéaire		Algorithme génétique					
	Solution optimale	Temps de calcul (s)	Solution optimale		Temps de calcul (s)		Solution optimale	
			Pire	Meilleur	Plus long	Plus court	Pire	Meilleure
1	1213,0	0,23	1221,0	1213,0	4,28	3,27	1%	0%
2	1498,0	0,33	1534,0	1498,0	4,86	3,48	2%	0%
3	951,0	0,06	962,0	951,0	5,18	3,48	1%	0%
4	1228,0	0,06	1229,0	1228,0	4,61	3,22	0%	0%
5	1097,0	0,09	1097,0	1097,0	12,61	8,09	0%	0%
6	617,0	0,36	619,0	617,0	28,28	21,37	0%	0%
7	648,0	0,08	649,0	648,0	17,44	10,71	0%	0%
8	819,0	0,06	830,0	819,0	16,14	11,67	1%	0%
9	685,0	0,08	685,0	685,0	29,94	22,56	0%	0%
10	88,0	0,59	99,0	88,0	25,89	20,11	13%	0%
11	189,0	1,05	216,0	189,0	31,20	20,29	14%	0%
12	125,0	3,00	138,0	125,0	30,45	19,32	10%	0%
13	115,0	2,28	115,0	115,0	33,23	24,85	0%	0%
14	138,0	1,87	148,0	138,0	30,93	23,69	7%	0%
15	59,0	5,23	71,0	59,0	30,68	21,95	20%	0%
16	96,0	5,23	116,0	96,0	29,52	20,65	21%	0%
17	72,0	12,18	95,0	72,0	36,52	24,34	32%	0%
18	584,0	0,58	584,0	584,0	29,51	20,13	0%	0%
19	517,0	0,65	520,0	517,0	28,25	20,44	1%	0%
20	41,0	1,73	72,0	41,0	32,58	23,15	76%	0%
21	100,0	7,78	104,0	100,0	32,71	22,55	4%	0%
22	137,0	1,78	144,0	137,0	27,25	19,22	5%	0%
23	66,0	1,11	67,0	66,0	29,24	22,63	2%	0%
24	280,0	0,61	292,0	280,0	32,72	19,17	4%	0%
25	200,0	0,14	202,0	200,0	25,40	19,29	1%	0%
26	295,0	0,48	302,0	295,0	23,27	14,42	2%	0%
27	235,0	0,16	237,0	235,0	23,34	18,60	1%	0%
28	102,0	2,57	106,0	102,0	33,06	21,24	4%	0%
29	68,0	0,97	77,0	68,0	30,76	24,25	13%	0%
30	519,0	0,41	519,0	519,0	27,75	18,74	0%	0%
31	611,0	0,42	615,0	611,0	23,15	14,76	1%	0%
32	565,0	0,34	565,0	565,0	22,98	18,19	0%	0%
33	661,0	0,14	661,0	661,0	21,53	14,57	0%	0%
34	164,0	1,09	174,0	164,0	30,70	18,06	6%	0%
35	193,0	0,50	219,0	193,0	27,77	18,75	13%	0%
36	548,0	0,17	548,0	548,0	17,87	12,16	0%	0%
37	523,0	0,53	523,0	523,0	34,30	24,66	0%	0%
38	632,0	0,09	637,0	632,0	15,98	13,21	1%	0%
39	795,0	0,09	801,0	795,0	15,91	10,35	1%	0%
40	154,0	0,81	167,0	154,0	29,26	19,57	8%	0%
41	199,0	0,19	202,0	199,0	31,58	20,91	2%	0%
42	537,0	0,41	537,0	537,0	18,46	14,84	0%	0%
43	569,0	0,50	569,0	569,0	23,05	19,00	0%	0%
44	556,0	0,59	556,0	556,0	18,35	11,58	0%	0%
45	760,0	0,11	760,0	760,0	17,20	13,19	0%	0%

Annexe C : Procédure de génération des données pour l'analyse numérique de la complexité

Afin d'analyser numériquement la complexité du programme linéaire (chapitre 5, section 5.4.2) et du programme dynamique (chapitre 6, section 6.5.3), nous considérons un cas de base issu de notre cas d'étude. Il est présenté dans le tableau C.1 (retard ou repos par véhicule par poste). Nous rappelons que les éléments négatifs correspondent à des repos. La ligne « Intervalle » indique le minimum et le maximum des retards (ou repos) pour chaque opérateur. Les données seront générées à partir du cas de base.

Tableau C.1 : Données du cas de base

Modèles	**Opérateur 1**	**Opérateur 2**	**Opérateur 3**	**Opérateur 4**	**nombre de produits par modèle**
m1	0,3	0,1	-2,2	-1,1	3
m2	0,1	1,3	-1,6	-0,7	5
m3	-0,5	0,3	-2,1	-1,1	5
m4	-1,1	-2,4	-3,1	1	4
m5	-1,1	-2,4	2	-1	3
Intervalle	[-1.1, 0.3]	[-2.4, 1.3]	[-3.1, 2]	[-1.1, 1]	
Taux de charge	95,9%,	96,1%,	85,8%	94,9%	

Pour l'analyse de l'effet du <u>taux de charge</u>, notre procédure de génération des données est décrite ci-dessous :

Dans le cas de base, on a 20 produits, 4 opérateurs et 5 modèles. Nous allons garder ces paramètres inchangés pour tester l'effet du taux de charge. Nous gardons aussi les nombres de produits par modèle du cas de base.

On souhaite augmenter progressivement le taux de charge moyen des opérateurs. On l'augmente de 1% à chaque fois et nous générons 4 instances avec le même taux de charge moyen. Notons *tx* le taux de charge voulu.

On définit des limites pour les taux de charge de chaque opérateur pour que les opérateurs ne soit ni en surcharge, ni en sous charge. Nous utilisons 99% comme taux maximal et 85% comme taux minimal.

Notons t_{inf} et t_{sup} les limites respectivement inférieure et supérieure des temps opératoires de tous les opérateurs dans le cas de base. Nous allons générer des temps opératoires sans dépasser ces limites pour que les temps soient raisonnables.

La procédure de génération est la suivante :

Répéter

Pour chaque opérateur faire

Répéter

Générer aléatoirement *5* temps opératoires avec une loi uniforme entre t_{inf} et t_{sup}

Calculer le taux de charge en multipliant les modèles par leurs nombres de produit

Jusqu'à taux de charge de l'opérateur appartient à [85%, 99%]

Fin pour

Jusqu'à taux de charge moyen des 4 opérateurs égal à *tx*

Pour l'analyse de l'effet du <u>nombre d'opérateurs</u>, notre procédure de génération des données est décrite ci-dessous :

Nous voulons augmenter progressivement le nombre d'opérateurs. Nous générons alors des temps opératoires pour de nouveaux opérateurs. Nous gardons les autres paramètres inchangés. On doit avoir 5 modèles, 20 produits et un taux de charge moyen de 92,5%. Nous générons 4 instances par nombre d'opérateurs.

Nous utilisons les limites de taux de charge et de temps opératoires utilisées dans la procédure précédente. La procédure de génération est la suivante :

Répéter

Pour chaque opérateur à générer faire

Répéter

Générer aléatoirement *5* temps opératoires avec une loi uniforme entre t_{inf} et t_{sup}

Calculer le taux de charge en multipliant les modèles par leurs nombres de produit

Jusqu'à taux de charge de l'opérateur appartient à [85%, 99%]

Fin pour

Jusqu'à taux de charge moyen de tous les opérateurs égal à *92,5%*

Pour l'analyse de l'effet du <u>nombre de modèles</u>, notre procédure de génération des données est décrite ci-dessous :

On souhaite augmenter progressivement le nombre de modèle en gardant le même nombre de produits, le même nombre d'opérateurs et le même taux de charge. Notons *M* le nombre de modèles voulu pour l'instance à générer et n_m le nombre de produits pour le modèle *m*. Nous limitons le nombre de produits par modèle à 5. La procédure de génération est la suivante :

1. Répéter

Générer aléatoirement *M* entiers avec une loi uniforme entre 1 et 5.

Jusqu'à avoir la somme des nombres de produits par modèle égale à 20.

2. Répéter

Pour chaque opérateur à générer faire

Répéter

Générer aléatoirement *M* temps opératoires avec une loi uniforme entre t_{inf} et t_{sup}

Calculer le taux de charge en multipliant les modèles par les nombres de produits n_m

Jusqu'à taux de charge de l'opérateur appartient à [85%, 99%]

Fin pour

Jusqu'à taux de charge moyen des tous les opérateurs égal à *92,5%*

Pour l'analyse de l'effet du <u>nombre de produits</u>, notre procédure de génération des données est décrite ci-dessous :

On souhaite augmenter progressivement le nombre de produits en gardant le même nombre de modèles, le même nombre d'opérateurs et le même taux de charge. Notons *N* le nombre de produits voulu pour l'instance à générer. Nous limitons le nombre de produits par modèle à 7.

La procédure de génération est la suivante :

Répéter

Générer aléatoirement *5* entiers avec une loi uniforme entre 1 et 7

Calculer le taux de charge en multipliant les modèles par leurs nombres de produit

Jusqu'à avoir (la somme des nombres de produits par modèle égale au N voulu) et (taux de charge moyen des tous les opérateurs égal à *92,5%*)

Annexe D : La performance de l'heuristique basée sur la programmation dynamique pour les tests du cas d'étude

Dans cette annexe, nous présentons les résultats des tests de l'heuristique basée sur la programmation dynamique pour les instances du cas d'étude (chapitre 6, section 6.6.2). Nous comparons les solutions trouvées à celles du programme linéaire pour les mêmes instances. On arrête le calcul au bout de 10 minutes. Le gap entre la solution trouvée par l'heuristique et celle du programme linéaire se calcule comme suit :

$$Gap = \frac{Solution\ Prog.Linéaire - Solution\ de\ l'heuristique}{Solution\ Prog.Linéaire}$$

Le tableau D.1 donne, pour chaque couple (CR_0, SR), la moyenne des gaps entre la solution trouvée par l'heuristique et celle du programme linéaire sur les neuf instances testées.

Tableau D.2 : Moyenne du gap de la solution de l'heuristique

CR_0 \ SR	0,1	0,2	0,3	0,4	0,5	0,6	0,7	0,8	0,9
0,1	98,4%	98,4%	98,4%	98,4%	98,4%	98,4%	98,4%	98,4%	98,4%
0,2	98,4%	98,4%	98,4%	98,4%	98,4%	98,4%	98,4%	98,4%	98,4%
0,3	68,2%	84,9%	97,4%	98,4%	98,4%	98,4%	98,4%	98,4%	98,4%
0,4	56,7%	63,0%	87,0%	97,6%	98,4%	98,4%	98,4%	98,4%	98,4%
0,5	54,1%	55,9%	69,9%	90,9%	97,6%	98,4%	98,4%	98,4%	98,4%
0,6	81,7%	59,0%	63,1%	75,0%	89,9%	97,6%	98,4%	98,4%	98,4%
0,7	89,0%	85,4%	86,1%	71,3%	77,7%	90,9%	96,5%	98,4%	98,4%
0,8	90,0%	89,2%	88,8%	89,0%	88,6%	84,0%	93,3%	96,0%	98,4%
0,9	94,3%	94,2%	94,2%	91,9%	90,8%	89,2%	90,8%	93,8%	97,6%
0,95	95,1%	95,1%	95,1%	95,0%	94,4%	94,2%	91,9%	92,6%	95,7%
0,97	95,1%	95,1%	95,1%	95,1%	95,1%	95,1%	94,4%	95,0%	95,3%
0,99	95,1%	95,1%	95,1%	95,1%	95,1%	95,1%	95,1%	95,1%	95,1%

Le tableau D.2 donne, pour chaque couple (CR_0, SR), le minimum des gaps entre la solution trouvée par l'heuristique et celle du programme linéaire sur les neuf instances testées.

Tableau D.2 : Minimum du gap de la solution de l'heuristique

CR_0 \ SR	0,1	0,2	0,3	0,4	0,5	0,6	0,7	0,8	0,9
0,1	29,3%	29,3%	29,3%	29,3%	29,3%	29,3%	29,3%	29,3%	29,3%
0,2	29,3%	29,3%	29,3%	29,3%	29,3%	29,3%	29,3%	29,3%	29,3%
0,3	29,3%	29,3%	29,3%	29,3%	29,3%	29,3%	29,3%	29,3%	29,3%
0,4	29,3%	29,3%	29,3%	29,3%	29,3%	29,3%	29,3%	29,3%	29,3%
0,5	20,5%	29,3%	29,3%	29,3%	29,3%	29,3%	29,3%	29,3%	29,3%
0,6	8,8%	14,9%	29,3%	29,3%	29,3%	29,3%	29,3%	29,3%	29,3%
0,7	29,3%	12,6%	21,8%	26,7%	29,3%	29,3%	29,3%	29,3%	29,3%
0,8	29,3%	29,3%	29,3%	29,3%	29,3%	29,3%	29,3%	29,3%	29,3%
0,9	29,3%	29,3%	29,3%	29,3%	29,3%	29,3%	29,3%	29,3%	29,3%
0,95	29,3%	29,3%	29,3%	29,3%	29,3%	29,3%	29,3%	29,3%	29,3%
0,97	29,3%	29,3%	29,3%	29,3%	29,3%	29,3%	29,3%	29,3%	29,3%
0,99	29,3%	29,3%	29,3%	29,3%	29,3%	29,3%	29,3%	29,3%	29,3%

Le tableau D.3 donne, pour chaque couple (CR_0, SR), le maximum des gaps entre la solution trouvée par l'heuristique et celle du programme linéaire sur les neuf instances testées.

Tableau D.2 : Maximum du gap de la solution de l'heuristique

CR_0 \ SR	0,1	0,2	0,3	0,4	0,5	0,6	0,7	0,8	0,9
0,1	191,9%	191,9%	191,9%	191,9%	191,9%	191,9%	191,9%	191,9%	191,9%
0,2	191,9%	191,9%	191,9%	191,9%	191,9%	191,9%	191,9%	191,9%	191,9%
0,3	98,9%	136,8%	191,9%	191,9%	191,9%	191,9%	191,9%	191,9%	191,9%
0,4	99,6%	94,6%	177,6%	191,9%	191,9%	191,9%	191,9%	191,9%	191,9%
0,5	109,6%	96,6%	119,8%	191,9%	191,9%	191,9%	191,9%	191,9%	191,9%
0,6	166,9%	129,6%	112,1%	156,2%	191,9%	191,9%	191,9%	191,9%	191,9%
0,7	191,9%	187,5%	191,9%	129,6%	177,6%	191,9%	191,9%	191,9%	191,9%
0,8	191,9%	191,9%	191,9%	191,9%	186,8%	191,9%	191,9%	191,9%	191,9%
0,9	191,9%	191,9%	191,9%	191,9%	191,9%	191,9%	191,9%	191,9%	191,9%
0,95	191,9%	191,9%	191,9%	191,9%	191,9%	191,9%	191,9%	191,9%	191,9%
0,97	191,9%	191,9%	191,9%	191,9%	191,9%	191,9%	191,9%	191,9%	191,9%
0,99	191,9%	191,9%	191,9%	191,9%	191,9%	191,9%	191,9%	191,9%	191,9%

Résumé

Dans cette thèse, nous considérons le problème du séquencement sur une ligne de montage multi-modèles de véhicules industriels. Pour équilibrer au mieux la charge dynamique des opérateurs, la minimisation de la somme des retards à l'issue de chaque véhicule est proposée.

Deux approches peuvent être utilisées pour optimiser le lissage de charge dans un problème de séquencement : l'utilisation directe des temps opératoires ou le respect de règles. La plupart des travaux appliqués à l'industrie automobile utilisent l'approche de respect de règles. Une originalité de ce travail est d'utiliser l'approche de la prise en compte directe des temps opératoires.

L'étude de la littérature de ce problème a dévoilé deux lacunes dans les travaux précédents : l'essentiel des travaux modélisent un seul type d'opérateurs d'une part, et proposent des heuristiques ou des métaheuristiques pour résoudre ces problèmes, d'autre part. L'originalité de ce travail est de tester des méthodes exactes pour des instances industrielles et de modéliser le fonctionnement de trois différents types d'opérateurs spécifiques au cas industriel.

Deux méthodes exactes sont développées : la programmation linéaire mixte et la programmation dynamique. Une étude expérimentale des facteurs de complexité sur des instances académiques des deux modèles est développée. Les modèles sont aussi testés sur des instances du cas d'étude.

Par ailleurs, le problème est traité par deux méthodes approchées : une heuristique basée sur la programmation dynamique d'une part, et des métaheuristiques (algorithme génétique, recuit simulé et un couplage des deux) d'autre part. Les deux approches sont testées sur des instances académiques et des instances du cas d'étude.

Ce travail a permis d'apporter une solution intéressante d'un point de vue industriel puisqu'il prend en compte les caractéristiques de la ligne de montage (opérateurs spécifiques) et améliore significativement la qualité du séquencement en un temps de calcul raisonnable.

Mots clés : ligne d'assemblage multi modèles, séquencement, minimisation des retards, programmation linéaire mixte, programmation dynamique, métaheuristiques.

Abstract

In this thesis, the problem of sequencing mixed model assembly lines (MMAL) is considered. Our goal is to determine the sequence of products to minimize the work overload. This problem is known as the mixed model assembly line sequencing problem with work overload minimization (MMSP-W). This work is based on an industrial case study of a truck assembly line.

Two approaches can be used to minimize the work overload: the use of task operation times or the respect of sequencing rules. Most of the earlier works applied in car industry use the latter approach. The originality of this work is to employ the task operation times for the generation of the product sequence in a MMAL.

The literature review has highlighted two main gaps in previous works: most of the papers consider a single type of operators, and propose heuristics or metaheuristics to solve the problem. The originality of this work is to test exact methods for industrial case instances and to model three different types of operators.

Two exact methods are developed: the mixed integer linear programming and dynamic programming. The models are tested on industrial case study instances. An experimental study is developed for both approaches in order to understand the complexity factors.

Moreover, the problem is treated by two approximate methods: a heuristic based on dynamic programming and metaheuristics (genetic algorithm, simulated annealing and a hybrid method based on both genetic algorithm and simulated annealing). All approaches are tested on academic instances and on real data from the industrial case study.

Keywords: Mixed model assembly line, sequencing, work overload minimization, mixed integer linear programming, dynamic programming, meta-heuristics.